CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Metrics for Maintainability of Executable Models

Bachelor of Science Thesis in Software Engineering and Management

Fisnik Hajredini
Sepehr Afrouzimanesh

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

FISNIK HAJREDINI
SEPEHR AFROUZIMANESH

Supervisor: JAN SCHRÖDER
Examiner: PIERGIUSEPPE MALLOZZI

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Metrics for Maintainability of Executable Models

Fisnik Hajredini
IT department
University of Gothenburg
Gothenburg, Sweden
gushajfi@student.gu.se

Sepehr Afrouzimanesh
IT department
University of Gothenburg
Gothenburg, Sweden
gusafrse@student.gu.se

***Abstract*—Model-based software development plays a key role in modern day industries and the size and complexity of it is increasing constantly. However far too little attention has been paid to maintainability quality characteristics of model-based software. In this paper we provide the state of art analysis throughout a systematic literature review and present the existing maintainability metrics of executable models. In addition, a survey is conducted to shed light on practitioners perspective on maintainability metrics in industry. Metrics compositions and their dependencies are visualized on a diagram. The relation between the metrics and the quality characteristics they try to resolve, is also presented as a part of our result. We conclude that more research is needed on metrics of this field, because model-based software is usually being practiced in closed-source communities.**

***Keywords*-Model based software; Executable model; Metric; Maintainability ; Quality; Complexity; Simulink; Measurement; Assessment; Mofifiability;**

I. Introduction

Nowadays Model-based software (MBS) is widely used within many industries, especially those which are conducting development of large embedded systems such as automotive and aerospace companies. For example, automotive companies such as Volvo Car Group, Audi AG, and similar manufactures primarily use MBS such as MATLAB/Simulink to develop logic and application for their systems. One significant aspect of their work is to develop high quality software which is essential to have high quality product. Each day more and more complex softwares and electronics are being developed in this area which makes continuous change and updates inevitable. This causes maintainability to be an important quality of the system due to increasing complexity and size of executable models and therefore maintenance is a necessity. The ability to observe software qualities like maintainability is a great importance and can be achieved by using metrics.

Measurement of qualities in software development environments impact the cost and development process. Putnam and Myers [1] highlight in their work, that if metrics are poorly chosen, inaccurately collected and unwisely applied, teams solve problems deficiently. They argue that metrics can provide time and staff allowance within the scope of project. In software, there are plentiful of well known product and process metrics that have been designed for object-oriented programming (OOP) and they are still being developed and adapted in their respective fields [2]. Many of these metrics are directly aiming maintainability as an important quality characteristic of object-oriented programs in order to improve, compare and evaluate. However there hasn't been enough studies on the quality aspects of MBS. After all, maintainability metrics are rare and insufficient and there is a gap in terms of metric dedicated to executable models.

Thus our research is aimed to find, gather and analyze MBS-related researches and the goal is to go through existing literatures and find metrics that measures maintainability of executable models. In additions, in this paper we propose industry perspective on our findings in order to have a broader view on this matter. To address this issue these question need to be answered:

- RQ1: What metrics exist to measure maintainability in executable models?
  - RQ1.1: How are existing maintainability metrics for executable models in line with practitioners perspective?

This paper presents our finding from a literature review with the aim to collect the state of art related to quality metrics, maintainability in particular, regarding executable models. In addition, by using the literature review findings, a survey is designed and conducted in automotive companies to find how maintainability metrics are perceived in industry.

The rest of the paper is structured as follows. In the Section 2 we describe and discuss existing similar literature which relates to this paper such as UML models metric and open-source metrics. Section 3 is about the methodology of this research including the research processes, data collection and analysis. In section 4 existing metrics on executable models are presented. In section 5 we visualized survey results. In section 6 Two figures are presented which the first one shows the relationship between base measures and measurement functions and the second one displays the relationship between measurement functions with quality characteristics and sub characteristics. Moreover, the relationship between our finding and industry perspective will be discussed. Section 7 is based on our final conclusion and possible future works that needs to be done and finally Section 8 is Acknowledgements.

II. Related Work

Since this papers methodology is literature review, we didn't find any other research that was conducted with the same subject and methodology as our paper. Therefore, we decided to consider related work the different paradigms of software engineering (besides model based software). Similar methods in two other paradigms were proposed that solve relatively similar problems: (1). UML models metrics and (2). open-source metrics (OOP).

We found that for UML models, counting the number of interactions in a class, can influence the coupling degree [3], this would be similar to the model-based software, where the number of signals can also be related to coupling. Besides metrics, Lange et al [4] developed a MetricViewEvolution which is a tool that collects some metrics from UML models and visualizes them.

On the other hand, for object oriented programming, there are already many studies conducted. For the size of a software or module, the easiest metric of size is the lines of code (LOC) which we also find on the model based software, as generated lines of code. Besides the size metric, we find coupling, cohesion and inheritance metrics that predict maintainability [5].

III. METHODOLOGY

We answer our main research question of existing metrics by reviewing existing literature on maintainability metrics for model-based software. Afterwards, based on the literature review results, we conduct a survey to answer our sub-research question regarding practitioners perspective on maintainability. We used a mixed method approach where we can use qualitative and quantitative data from the reviewed literature and quantitative data from survey.

A. Literature review

The literature review is conducted following the guidelines of B.Kitchenham [6]. Accordingly, first, a search string was designed and 4 databases were selected to conduct the search which resulted in 419 papers. Afterwards a systematic review was conducted on all the papers. Our systematic review contains inclusion/exclusion criteria, data extraction and data synthesis, which are all included on Kitchenham procedures.

- **Search string**
  Our search string is composed of 12 strings, that is because some databases limited the number of strings that can be used. The strings were derived based on an approach from B.Kitchenham [6]. The approach includes Population, Intervention and Outcomes (cf. Fig. 2). Our population includes targeted areas, while our intervention is the action to be taken. Finally our outcome is the goal of the action on the targeted area. The search is conducted using the search string on IEEExplore, ACM Digital library, Science Direct, and Engineering Village which resulted in a total of 419 studies.
  ("Model based software" OR "Executable models" OR Simulink OR Dymola OR Modelica) AND (Metrics OR Measurements OR Assessment) AND (Maintainability OR Modifiability OR Complexity OR Quality)
- **Inclusion and exclusion**
  We designed inclusion/exclusion criteria for this process so that both researchers can refer to it when excluding a paper. Our inclusion/exclusion criteria aims to select papers which can answer our main research question regarding existing metrics. Accordingly, our inclusion/exclusion criteria defines our study scope. During this process we excluded papers that were not in English. We also excluded any papers that have fewer than two pages to ensure quality of the papers. Papers that were not related to model based software, but still mention the use of Simulink, were also excluded. These cases didn't correspond with our criteria below and therefore were excluded. Selected final papers are the result of our inclusion and exclusion criteria.
  - Inclusion criteria
    * The paper must be in English
    * The paper must be longer than 2 pages
    * The paper must discuss the maintainability of model based software
    * The paper must discuss quality metrics of model based software
  - Exclusion criteria
    * Papers that mention quality or complexity but don't refer directly to model based software
    * Papers that do not discuss or mention the measurement of quality
    * Papers that mention only measurements of functionality, like functional testing or reliability, for example
    * Papers that are not software related

  In order to perform the inclusion/exclusion procedure efficiently we created a form to list all papers. Once a paper is reviewed, we would mark it with a Green (pass to next phase), Yellow (unsure), and Red (not related to our research). All the papers were reviewed from each author and when a paper resulted in combination of Yellow and Red, a third reviewer assisted with a review. Any green combination passed to the next phase and two yellow votes passed to the next phase as well. Our inclusion/exclusion procedure was performed in three iterations (cf. Fig. 1). First, by reading title and abstract, we excluded most of the papers. Of the remaining papers, we read the introduction and conclusion sections where we excluded 22 additional papers. Finally, a full text review was conducted right before data extraction which left us with 8 final papers [2] [7] [8] [9] [10] [11] [12] [13].
  This paper defines maintainability quality characteristic as ISO 25010 standard, with its sub-characteristics, Modification stability, Changeability, Testability, Analysability, Modularity, Maintainability compliance, and Re-usability. Some of the papers have been published before the ISO 25010 standard, therefore there are qualities that do not match with our definition. Nevertheless, we include metrics that measure other quality characteristics such as understandability, adaptability, modifiability, and scalability because they might lead to complexity measurements.

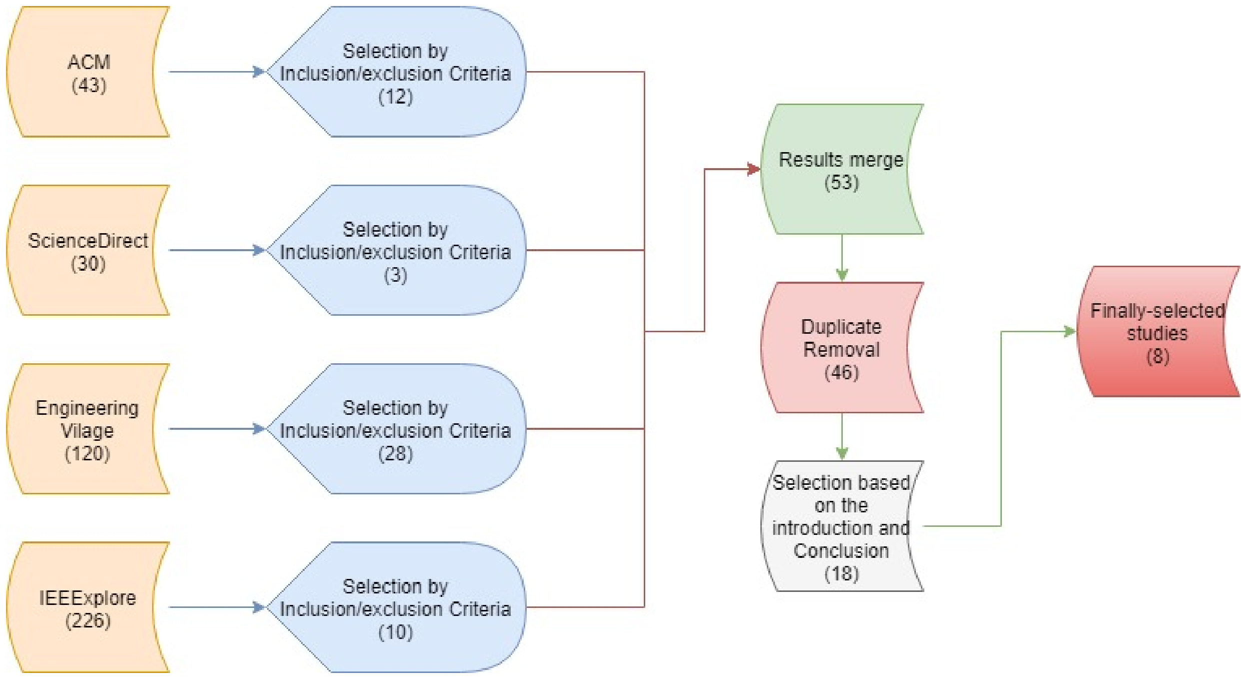
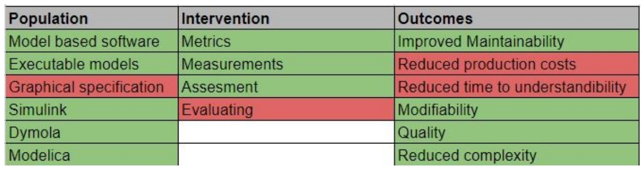

ACM (43)
ScienceDirect (30)
Engineering Vilage (120)
IEEExplore (226)
Selection by Inclusion/exclusion Criteria (12)
Selection by Inclusion/exclusion Criteria (3)
Selection by Inclusion/exclusion Criteria (28)
Selection by Inclusion/exclusion Criteria (10)
Results merge (53)
Duplicate Removal (46)
Selection based on the introduction and Conclusion (18)
Finally-selected studies (8)

Fig. 1: Roadmap to final papers



Population	Intervention	Outcomes
Model based software	Metrics	Improved Maintainability
Executable models	Measurements	Reduced production costs
Graphical specification	Assesment	Reduced time to understandibility
Simulink	Evaluating	Modifiability
Dymola		Quality
Modelica		Reduced complexity

Fig. 2: Search string definition

- **Data extraction** We designed a form were we initially extracted the following general information:
  - Name of Reviewer
  - Date of Data extraction
  - Title, authors, journal, publication details
  - Space for additional notes

  Besides the general information, our focus was on data that answers our research question and data that can be synthesized for the next procedure. For example, we collect the metrics mentioned in the studies and their description to answer our main research question. Furthermore, we collect the population of the data used in the studies to investigate the domain where the metrics are applied. Therefore we extracted the following:
  - Number of metrics
  - Population description
  - Underlying attributes of the metric goal
  - List of metrics
  - Description of the metrics
  - Outcome of the metrics

  During our data extraction of a particular paper [8], we had trouble extracting information about the description and outcome of the metrics. Therefore we contacted the authors of the paper and acquired the information. One of the authors responded and the description of the metric together with the outcome were added to our form.
- **Data synthesis**
  Based on our findings, we divide the metrics as base measurements and measurement functions [14]. For our synthesis, we conduct a qualitative and quantitative approach. We visualize the relation between base measurements and measurement functions on a diagram together with the papers references that mention them. Furthermore, we discuss metrics that were homogeneous with other papers and the qualities that the metrics were aiming to measure.

B. *Survey*

Once the metrics from all our papers were extracted, we started designing the survey to answer our sub-research question. Since automotive makers are one of the industries that experience metrics limitation on maintainability of model based software, we conducted a survey on their practitioners of Simulink within the automotive industry.

The survey was sent to engineering staff, team leaders and some managers. Because of the limited use of models-based software even within the automotive industry we decided to base our sampling on convenience. Via email, We asked our personal contacts, known to work with executable models, to answer the survey and spread it further. The survey was online and 7 answers were received, 6 of which were valid. Initially we gather basic information such as the country of participant, chosen survey language, time of the survey started, education and Job position. Besides the standard data, we gather the modeling domain, modeling languages, and their experience on the modeling language. Once the basic data is received, we focus on questions regarding our found metrics.

Survey was designed with 25 questions. Participants were asked to decide the impact of each base measure(factor) on how easy a model can be understood or modified. We chose Likert-type scales for answering survey's questions (From 1 being much easier to modify, 2-easier to modify, 3-slightly easier to modify, 4-no influence, 5-slightly more difficult to modify, 6-more difficult to modify, 7-much more difficult to modify and additional check box for "I do not know"). This is because we wanted to see the level of their agreement with the found metrics.

IV. Existing maintainability metrics

Below we present the metrics that were extracted from 8 of the papers reviewed.

- **Complexity**
  This metric is based on general complexity model calculation created by Card and Glass 1990. Because Card and Glass's metric aims to measure systems based on modules, Marta Plaska and Marina Walden [7] adjust it to work for measuring model based systems complexity. They define complexity(C) as the sum of **Data complexity(D) and Structural complexity(S)**. Structural complexity is defined as following:
  $$S = \frac{\sum f^2(i)}{n}$$
  where $f(i)$ is fan-out of subsystem block i and n is a number of subsystem blocks in the system.
  $$D = \frac{V(i)}{n \cdot ((f(i) + 1))}$$
  Where $V(i)$ is the number of Input/Output variables in a subsystem block i,$f(i)$ and n are as above.
  This complexity metric is further mentioned and tested in other papers such as Schroeder et al [12], Olszewskas' further work [9], Olszewska et al [10],Hu et al [8]
- **Model size(Number of discrete states)**
  A model size is defined by the number of discrete states on the model. Its proposed from [7] and they use it to compare two applications differences.
- **Number of blocks**
  This metric defines the number of blocks in a subsystem or a system. All the 8 paper from our literature review relate the number of blocks to maintainability. They either compose measurement function, or use it as a sole metric [12] [8] [7]
- **The number of Goto/From block pairs**
  Number of connections between two blocks implemented with Goto/From blocks instead of lines visually connecting the blocks.From and Goto blocks allow you to pass a signal from one block to another without actually connecting them.This causes issues in readability, since it is not immediately clear which Blocks are connected.Hu et al [8] present this on a table as a sample metric
- **Lines of code**
  The number of generated lines of code from the executable model can indicate the size of the model. This is used as a metric from Schroder et al [12] and Plaska [7].
- **Generated code measurements**
  Plaska [7] proposes this metric to measure the size of the model, it includes all the measurements of generated code and can be further divided into measurements such as:
  – Number of lines of code
  – number of characters
  – number of words
  – number of characters without spaces
- **Compilation statistics**
  Plaska [7] uses simulink compilation statistics from to obtain information about the memory being used during the compilation and the models' time of compilation.
- **Information flow complexity**
  For the Simulink information flow complexity, Olszewska et al [10] defined the following metrics based on the Henry-Kafura's metrics.
  $$hkCMX = size \cdot (fanin \cdot fanout)^2$$
  where hkCMX is the information flow complexity of a sub- system, size is the number of contained blocks (including subsystem blocks), fan-in and fan-out represent the number of afferent and efferent blocks of a subsystem, respectively.
- **Ratio**
  Plaska [7] documents the ratio between the number of generated lines of code and the blocks.She uses the ratio to measure the size of a system and its complexity.
- **Library usage**
  Plaska [7] documents the use of libraries and its number. She claims that a library block increases the flexibility and portability of the system as updating the block automatically updates the system.
- **The ratio between referenced models and subsystems**
  The number of referenced models inside a block(similar like linking a library) can affect adaptability of a model. Hu et al [8] present this on a table as a sample metric .
- **Average slice size per input signal**
  A signal slice represents everything in the model the signal influences by its behavior or everything that influences the signals values, depending on the slicing direction. The average slice size can therefore be a general measure for the impact single signals can have or the dependency between different regions of the model. Hu et al [8] present this on a table as a sample metric and aim to measure scalability.
- **Instability**
  Instability(I) [9], [10] of a (subsystem) block is defined as the number of efferent couplings between blocks divided by the sum of afferent and efferent couplings between

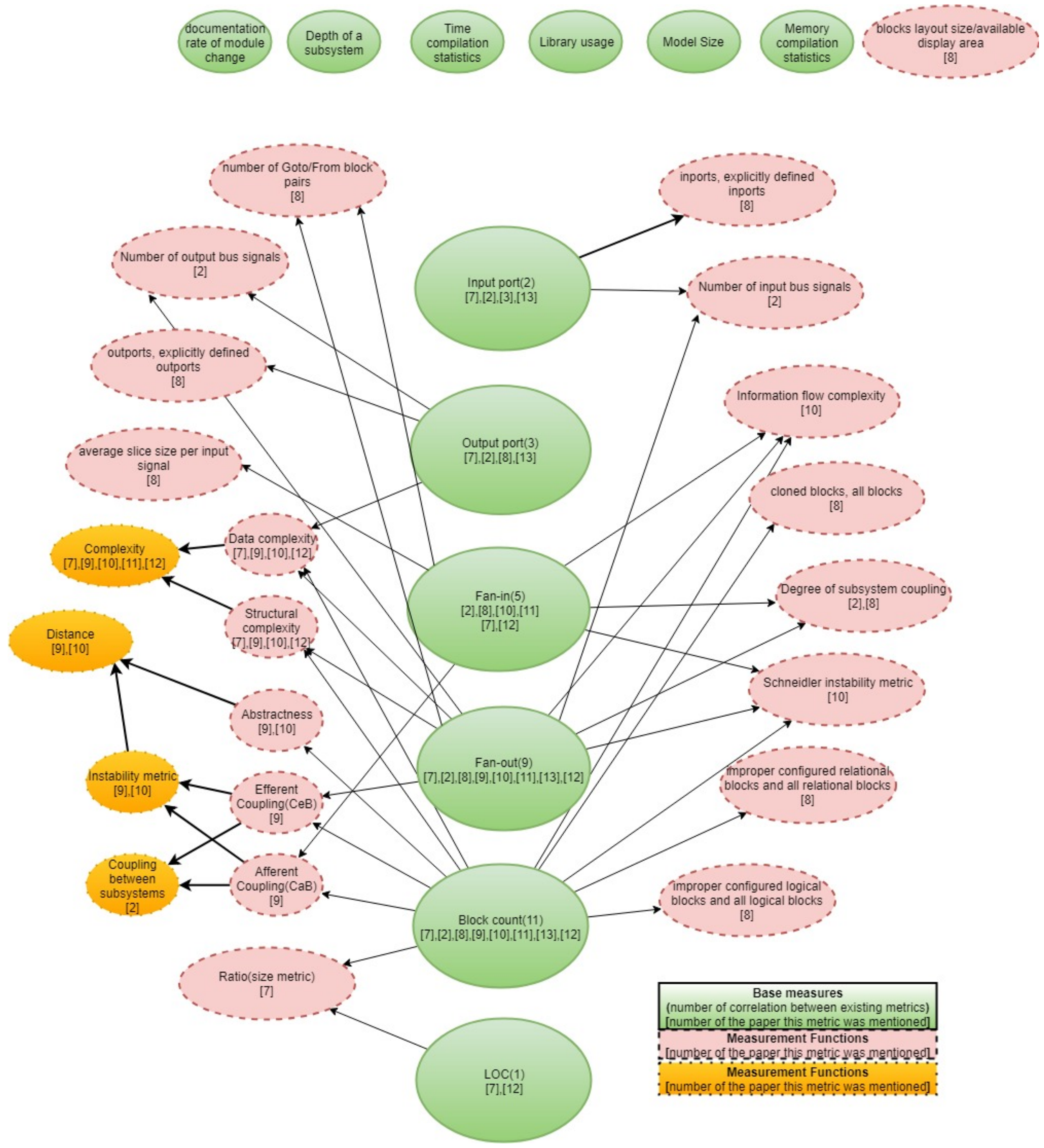

documentation rate of module change
Depth of a subsystem
Time compilation statistics
Library usage
Model Size
Memory compilation statistics
blocks layout size/available display area [8]
number of Goto/From block pairs [8]
inports, explicitly defined inports [8]
Number of output bus signals [2]
Input port(2) [7],[2],[3],[13]
Number of input bus signals [2]
outports, explicitly defined outports [8]
Information flow complexity [10]
average slice size per input signal [8]
Output port(3) [7],[2],[8],[13]
cloned blocks, all blocks [8]
Data complexity [7],[9],[10],[12]
Complexity [7],[9],[10],[11],[12]
Fan-in(5) [2],[8],[10],[11] [7],[12]
Degree of subsystem coupling [2],[8]
Structural complexity [7],[9],[10],[12]
Distance [9],[10]
Schneidler instability metric [10]
Abstractness [9],[10]
Fan-out(9) [7],[2],[8],[9],[10],[11],[13],[12]
improper configured relational blocks and all relational blocks [8]
Instability metric [9],[10]
Efferent Coupling(CeB) [9]
Coupling between subsystems [2]
Afferent Coupling(CaB) [9]
Block count(11) [7],[2],[8],[9],[10],[11],[13],[12]
improper configured logical blocks and all logical blocks [8]
Ratio(size metric) [7]
LOC(1) [7],[12]
Base measures (number of correlation between existing metrics) [number of the paper this metric was mentioned]
Measurement Functions [number of the paper this metric was mentioned]
Measurement Functions [number of the paper this metric was mentioned]

Fig. 3: Base measures relations with measurement functions.

blocks, which is given by the equation:

$$I = \frac{CeB}{CeB + CaB}$$

- **Distance**
  There is a relationship between instability and abstractness as OOP and it is defined as Distance(D). D is computed as a normalised sum of these values decreased by 1, which is given by the formula:

$$D = |D + I - 1|$$

- **Abstractness**
  abstractness(A) [9], [10] of a block is defined as a ratio of the number of contained abstract (i.e. subsystem) blocks (NaB) to the total number of blocks in a layer the given

block represents (NB), which is given by formula:

$$A = \frac{NaB}{NB}$$

The presented metric has the range <0..1>, where 0 denotes a concrete block and 1 represents a completely abstract block.

- **Number of input/output and output/input bus signals** is the difference between the number of input signals (NiS) and the number of input ports (NiP).
- **Degree of subsystem coupling**
Degree of Subsystem Coupling (DSC) [2] is defined to give additional weight to the output which imply more complexity.

$$DSC = W_i N_{is} + W_o N_{no}$$

where Wi = 1, is weight of input dependencies and Wo = 2, is weight of output dependencies
- **Coupling between subsystems**
To measure coupling between subsystems (CBS) [2], [8] for a given subsystem we count the number of subsystems coupled to the subsystem, i.e., receiving input signals from the subsystem or sending output signals to the subsystem
- **Depth of a subsystem**
(DoS) [2] is the maximum level the subsystem has till its basic subsystems. It is consider as cohesion metric.
- **Documentation rate of module change**
The rate of documentation update, per module change [8].
- **Improper configured logical blocks and all logical blocks**
Operational blocks in Simulink can be split into those that perform numerical operations (such as a Sum-block or a Product-block) and those that perform Logical operations, i.e. those that perform boolean operations (e.g. AND, OR, etc.). The Logical Operator block performs the specified logical operation on its inputs. An input value is TRUE (1) if it is nonzero and FALSE (0) if it is zero [8].
- **Ratio between blocks layout size and avaialble display area**
The size of the layout of a level in the model can be measured by the size of the blocks and the length of the lines connecting them, resulting in pixel dimensions for the model (i.e. the coordinates of the outermost artifacts of the model).
If this size is larger than the display size, the model has to be scaled down in order to get an overview, which in turn makes it harder to observe details of the model [8].
- **Improper configured relational blocks and all relational blocks** There are mixed blocks such as the RelationalOperator (e.g. "<") that compares to numerical signals and returns a boolean result [8].
- **The ratio between number of explicitly defined inports and number of all inports** The ratio between number of explicitly defined inports and number of all inports [8].
- **The ratio between number of explicitly defined outports and number of all outports**
The ratio between number of explicitly defined outports and number of all outports [8].
- **The ratio between cloned blocks and all blocks** The ratio between cloned blocks and all blocks [8].
- **Afferent Coupling**
Afferent coupling between blocks (CaB) is defined as measure of the total number of external blocks linked to a given block due to incoming signal within one layer. In other words, it is the number of destination blocks for the block under analysis [2].
- **Efferent Coupling**
Efferent coupling between blocks (CeB) is defined as the number of blocks that are linked to a given block due to outgoing signal within one layer, i.e. it is the number of source blocks for the given block [2].
- **Schneidler instability metric**
This metric [10] calculates the average stability of the blocks of a Simulink model and is based on the concepts of blocks and their fan-in and fan-out. It is indicated that a block with more fan-in blocks as fan-out blocks has a higher probability of change.
- **NTM (Number of Transitions)**
The number of transitions in (M) [11], that is expressed as Transition Space (TS) such that:

$$TS = \{t \mid t : B_i -^{x} /_{y} \rightarrow B_j\} \text{ where } B_i,$$

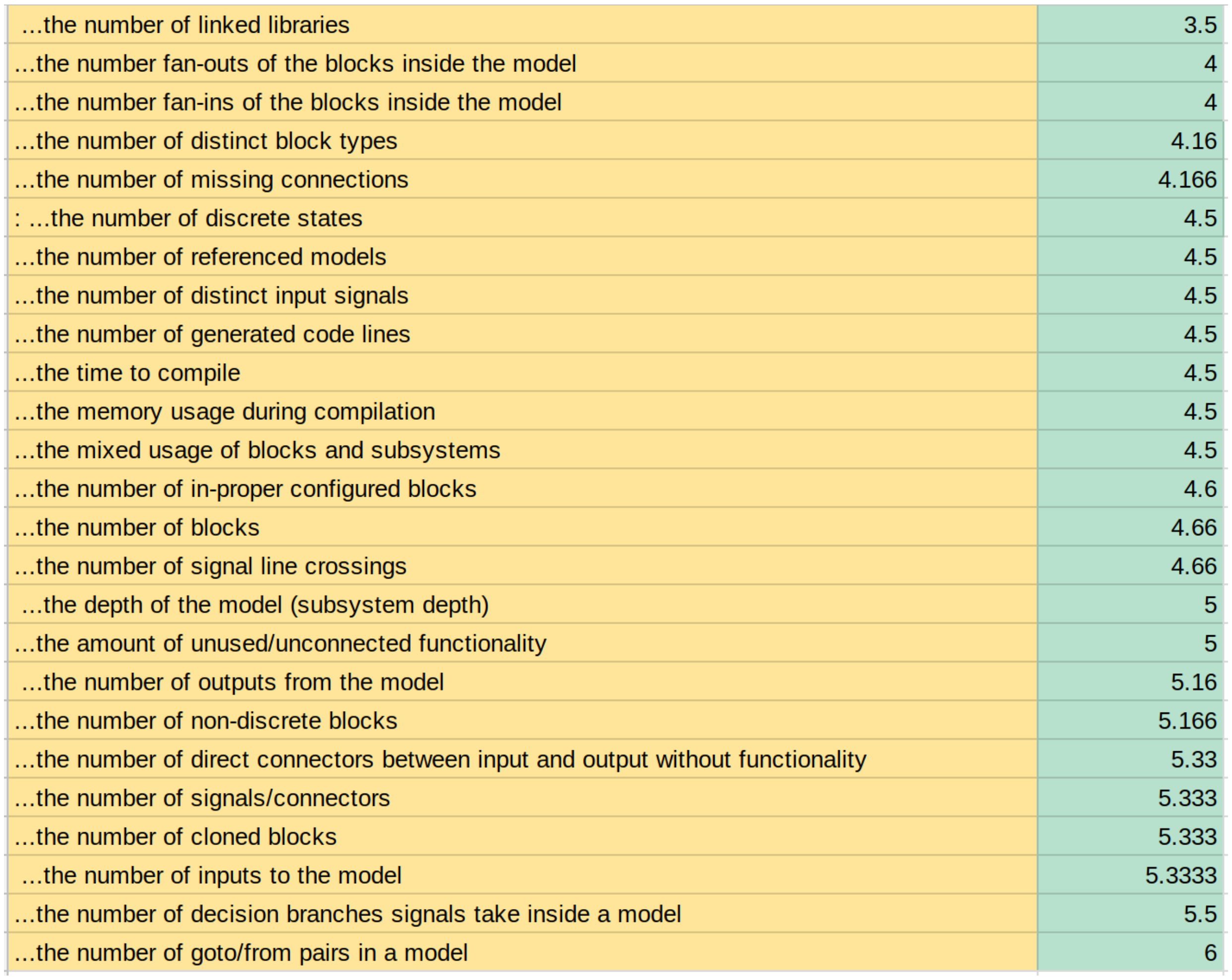
$$B_j \in E, x \in X, y \in Y, \mid TS \mid \leq k \times m$$

(Note that E: the set of blocks, where B0 is the initial block, such that |E|= k)
- **Fan-in**
The fan-in [11] of the block ei is the number of incoming transitions of ei arriving from another block ej where ei is a member of E.
- **Fan-out**
The fan-out [11] of the block ei is the number of outgoing transitions of ei towards another block ej where ei is a member of E.
- **MC (Model Complexity)**
Model Complexity [11] is the simplicity degree of relationships between blocks of the model at any level of hierarchy in M.

$$MC = \sum_{k=1}^{n} p_k * (fanin_k * fanout_k)^2$$

After analysis of the found metrics, 12 base measures were extracted, six of which composed 17 measurement functions(cf. Fig. 3). Both, measurement functions and base measurements, are used to measure maintainability or some sub-characteristics of maintainability in the collected papers.

Throughout data extraction we have categorized metrics, based on Staron and Meding [14], into two sub categories: Base measurements and Measurement functions. Base measurements are derived based on individual and basic attributes of an entity



…the number of linked libraries	3.5
…the number fan-outs of the blocks inside the model	4
…the number fan-ins of the blocks inside the model	4
…the number of distinct block types	4.16
…the number of missing connections	4.166
: …the number of discrete states	4.5
…the number of referenced models	4.5
…the number of distinct input signals	4.5
…the number of generated code lines	4.5
…the time to compile	4.5
…the memory usage during compilation	4.5
…the mixed usage of blocks and subsystems	4.5
…the number of in-proper configured blocks	4.6
…the number of blocks	4.66
…the number of signal line crossings	4.66
…the depth of the model (subsystem depth)	5
…the amount of unused/unconnected functionality	5
…the number of outputs from the model	5.16
…the number of non-discrete blocks	5.166
…the number of direct connectors between input and output without functionality	5.33
…the number of signals/connectors	5.333
…the number of cloned blocks	5.333
…the number of inputs to the model	5.3333
…the number of decision branches signals take inside a model	5.5
…the number of goto/from pairs in a model	6

Fig. 4: Answers of the survey based on a rating from 1 to 7.

in the system and measurement functions are algorithms that combines two or more base measures.

V. Practitioners rating

Based on the 6 valid results that we received from model based practitioners, we can evaluate existing metrics and see which base measure indicates a higher complexity or more difficult maintainability rate. All responders were from Germany and they were developers within automotive industry with educations such as mechatronics, electrical engineers and computer scientists. All the practitioners were using Simulink and didn't have any experience on the other modeling languages.

Another table was designed from replies on the metrics so that the reader can clearly understand the answers. They are in scale from 1 to 7, with 4 being neutral, 7 imposing highest influence on maintainability, and 1 imposing the lowest influence. On Figure 4, we can see the highest and lowest. The table shows the average value of of the answered questions.

On the first observation, we can see that the number of linked libraries has the lowest score of 3.5 from all base measures. Similarly, the number of fanout/fanin of blocks within the model was just on 4, however, only 2 of them rated the metric, the other 4 responders replied "I don't know". The number of missing connection also has quite little influence based on the practitioners, due to its score of 4.16 out of 7.

On the other hand, the peaks are more obvious since most of our answers were higher than the average, 4. We see a peak on the number of goto/from pairs in a model with an average of 6. The number of decision branches signals take inside the model scores 5.5, as the second most important metric. The number of signals/connectors, the number of direct connectors between input and output without functionality, the number of outputs, and the number of cloned blocks have all a value of 5.33.

Some honorable mentions are the depth of the model with a value of 5, the number of non discrete blocks with value of 5.16 and the amount of unused/unconnected functionalities with a value of 5.

VI. Discussion

Three different patterns were found after data synthesis of the study aim. Six of our papers [2] [7] [8] [9] [10] [11] were presenting metrics, five of the papers [7] [9] [10] [11]

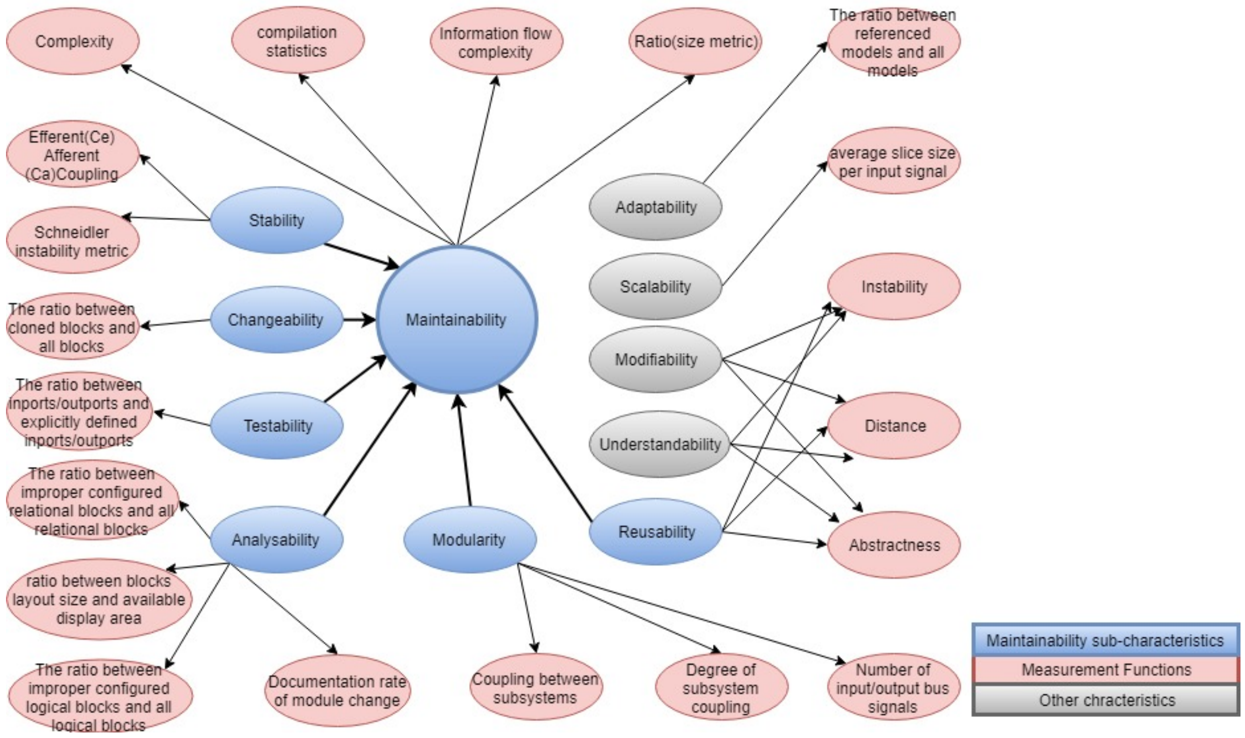

Complexity
compilation statistics
Information flow complexity
Ratio(size metric)
The ratio between referenced models and all models
Efferent(Ce) Afferent (Ca)Coupling
average slice size per input signal
Schneidler instability metric
Stability
Adaptability
Scalability
Instability
The ratio between cloned blocks and all blocks
Changeability
Maintainability
Modifiability
The ratio between inports/outports and explicitly defined inports/outports
Testability
Understandability
Distance
The ratio between improper configured relational blocks and all relational blocks
Analysability
Modularity
Reusability
Abstractness
ratio between blocks layout size and available display area
The ratio between improper configured logical blocks and all logical blocks
Documentation rate of module change
Coupling between subsystems
Degree of subsystem coupling
Number of input/output bus signals
Maintainability sub-characteristics
Measurement Functions
Other chracteristics

Fig. 5: Function measures and quality characteristic relation.

[12] were evaluating metrics and two of the papers [2] [13] presented tools to measure maintainability of model based software. Besides the aim of the study, it was interesting to observe the applicability population of the metrics. We found that 2 papers aimed to measure quality of hydraulic models, 3 aimed for automotive models, and other 3 aimed for different areas as long as its model based software.

Although Modelica and Dymola was in our search string, we did not get any final result that was discussing maintainability and was tested on its models. All the papers received were aimed or tested on Simulink models.

We dissolved the measurement functions, in order to relate them to their base measures. We discovered that 6 of our base measures composed all the other measurement functions (cf. Fig. 3). In cases such as Complexity or Distance we also find that they are composed of other measurement functions. On the other hand, we find 6 other metrics such as Library usage, or memory compilation statistics that are base measures which don't compose any measurement function. Its important to note that in some papers, base measurements were also aiming to measure maintainability or some sub-characteristics of maintainability.

Besides the relation between measurement functions and base measures, we visualized the quality characteristics that the metrics aim to resolve (cf. Fig. 5). However, on a few of the characteristics, they use old standard definitions to define maintainability. The qualities are still displayed but they are not a branch of maintainability based on ISO 25010. On a few papers, the definition of the precise quality the metric aims to measure is not exactly clear. Therefore we relate it only to maintainability but it might resolve other sub-characteristics as well.

On the 6 base measures that compose other function measures, we defined the number of links for each base measure. We observed that block count composed the most measurement functions and it was also mentioned in every reviewed paper. However, the result of survey suggests that block count doesn't influence maintainability on the same scale as other metrics, the survey participants gave an average of 4.66 which is slightly influential. Similarly, fan-out composes 9 other base metrics (2 less than block count) and is also mentioned in all papers. But whats interesting on the survey, 4 of the participants answered "I don't know" and 2 of them answered 4 (average). This might be a misunderstanding of the question posed as the same exact answers were received for fan-in.

An interesting perspective is that the hardest paper to have its data extracted [8] due to its lack of metrics description, showed the highest influence results on the practitioners answers. Most of the metrics presented there, such as the number of goto/from block, where only mentioned from Hu et al. This indicates that maintainability metrics in model based software, does indeed lack further research from Academia.

Although the low number of results from our survey is a validity threat, the clear distinction of results between academia and industry indicates that more research is necessary in this area.

Besides the low number of results from our survey, we consider the question on fan-in and fan-out as a validity threat as well. For our literature review, a higher number of papers to be reviewed by including more libraries, can enhance the quality and the number of papers found.

VII. Conclusion

Modern day industries have a great interest on the model based software development. As an example, automotive industries primarily design their systems in model-based software. With that being said, their system models needs to be frequently changing due to new technology adjustments and improvements. These changes often affect the size and complexity of the model which makes it more difficult to maintain. In other programing paradigms such as object oriented programing or UML models there are already metrics on maintainability [3] [5].

During our literature review, we found 419 papers which led us to extracting existing metrics on 8 different papers and then we compare our finding with the result of the survey that we have conducted with six industry practitioners. We visualized existing metrics on two diagrams, one which describes the relation between base measures and measurement functions, and the other defining the quality characteristics that measurement functions aim to measures. In addition to metric diagrams, we present a chart consisting the result of our survey.

The comparison between results led us to conclude that more research needs to be done on academia regarding maintainability metrics on executable models.

Acknowledgment

We thank University of Gothenburg for providing us access to the libraries where we managed to read and download the papers without having to pay for them. Besides the university of Gothenburg, special thanks to our supervisor Jan Schroeder for giving us constant feedback and guiding us through the thesis journey. We also appreciate the effort of Gideon Neumann to respond to our email regarding his paper [8]. We believe the participants added great value to our research as well answering the survey.

References

[1] L. H. Putnam and W. Myers, *Five core metrics: The Intelligence Behind Successful Software Management*. DORSET HOUSE PUBLISHING, 2003.

[2] Y. Dajsuren, M. G. van den Brand, A. Serebrenik, and S. Roubtsov, "Simulink models are also software," *Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures - QoSA '13*, p. 99, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2465478.2465482

[3] R. Harrison, S. Counsell, and R. Nithi, "Coupling metrics for object-oriented design," in *Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No.98TB100262)*, Nov 1998, pp. 150–157.

[4] C. F. J. Lange, M. A. M. Wijns, and M. R. V. Chaudron, "Metricviewevolution: Uml-based views for monitoring model evolution and quality," in *11th European Conference on Software Maintenance and Reengineering (CSMR'07)*, March 2007, pp. 327–328.

[5] M. Dagpinar and J. H. Jahnke, "Predicting maintainability with object-oriented metrics - An empirical comparison," *Proceedings - Working Conference on Reverse Engineering, WCRE*, vol. 2003-January, pp. 155–164, 2003.

[6] B. Kitcheham, *Procedures for Performing Systematic Reviews*, 2004.

[7] M. Plaska and M. Walden, "Quality comparison title of the technical and report evaluation of digital hydraulic control systems," no. 857, 2007.

[8] W. Hu, T. Loeffler, and J. Wegener, "Quality Model based on ISO / lEe 9126 for Internal Quality of MATLAB / SimulinkiStateflow Models," pp. 325–330, 2012.

[9] M. Olszewska, "Simulink-Specific Design Quality Metrics," no. August, 2011.

[10] M. Olszewska, Y. Dajsuren, H. Altinger, A. Serebrenik, M. Waldén, and M. G. J. van den Brand, "Tailoring complexity metrics for simulink models," *Proccedings of the 10th European Conference on Software Architecture Workshops - ECSAW '16*, pp. 1–7, 2016. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2993412.3004853

[11] E. A. Antonio, F. Ferrari, F. A. d. P. Caurin, and S. C. P. F. Fabbri, "A set of metrics for characterizing simulink model comprehension," *Journal of Computer Science & Technology*, vol. 14, no. 2, pp. 88–94, 2014. [Online]. Available: http://sedici.unlp.edu.ar/handle/10915/41807

[12] J. Schroeder, C. Berger, T. Herpel, and M. Staron, "Comparing the Applicability of Complexity Measurements for Simulink Models during Integration Testing - An Industrial Case Study," *Proceedings - 2nd International Workshop on Software Architecture and Metrics, SAM 2015*, pp. 35–40, 2015.

[13] "Extensible and automated model-evaluations with INProVE," vol. 6598 LNCS, 2011, pp. 193–208.

[14] M. Staron and W. Meding, "Industrial self-healing measurement systems," *Continuous software engineering*, vol. 9783319112831, no. August 2014, pp. 183–200, 2014.