Department of Electrical Engineering
An-Najah National University
(Nablus,Palestine)

Graduation Project

Validation of Model Metrics as Indicators on Model Quality

Author:
Omar J.S. Ayoub

Supervisor:
Pascal Montag
Jammal Khrousheh

DAIMLER

Stuttgart, Germany
May, 2012

Abstract

This study is an empirical investigation of the relations between model measurements and model quality. The main objective of this study is to define a set of model metrics which correlate strongly with failure reports and experts opinion. Then, to use these metrics as indicators on model quality and predictors of model failure.

This study targets a group of 'Simulink' models used in the development of embedded software within the automotive industry. Model Based Development has been a fundamental method in the development of embedded software in the automotive industry for the last two decades.

Alongside empirical values, this study distinguish between all factors and variables which may effect the quality of a model as an influence network. By identifying the influential factors it will be possible to prevent negative development.

Acknowledgments

It is a pleasure to thank those who have made this thesis possible: *Dr. Allam Mousa* (An-Najah National University), *Dr. Ingo Kreutz* (Daimler), *Dr. Steffen Goertzig* (Daimler), *Pascal Montag* (Daimler), and *Jamal Khrousheh* (An-Najah National University), who have offered me supervision, guidance, and support from the initial to the final phases, enabling the development of this project. Additionally, I would like to thank *Dr. Jan Sheible*, for his willingness and collaboration.

I would like to show my gratitude to *Daniela Jorge* for her support in the editing and formatting process. Lastly, it is an honor for me to take this opportunity to thank my parents, *Rana* and *Jehad*, for their support and encouragement.

Omar J.S. Ayoub

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter seeks to explain fundamental concepts and calculations alongside this study. The research focuses on model quality, therefore, it is important to answer several key questions: Why is model quality a critical issue within the automotive industry? What do we mean by "quality assurance"? What are the methods and techniques used by developers to enhance the quality of products? How does this research address its main objective of aiding in improving work quality while reducing time and cost within the development cycle?

The last section of this chapter introduces metrics and models used to build the numerical database, the measurements which were normalized and scaled, and the general preparation of data within this study.

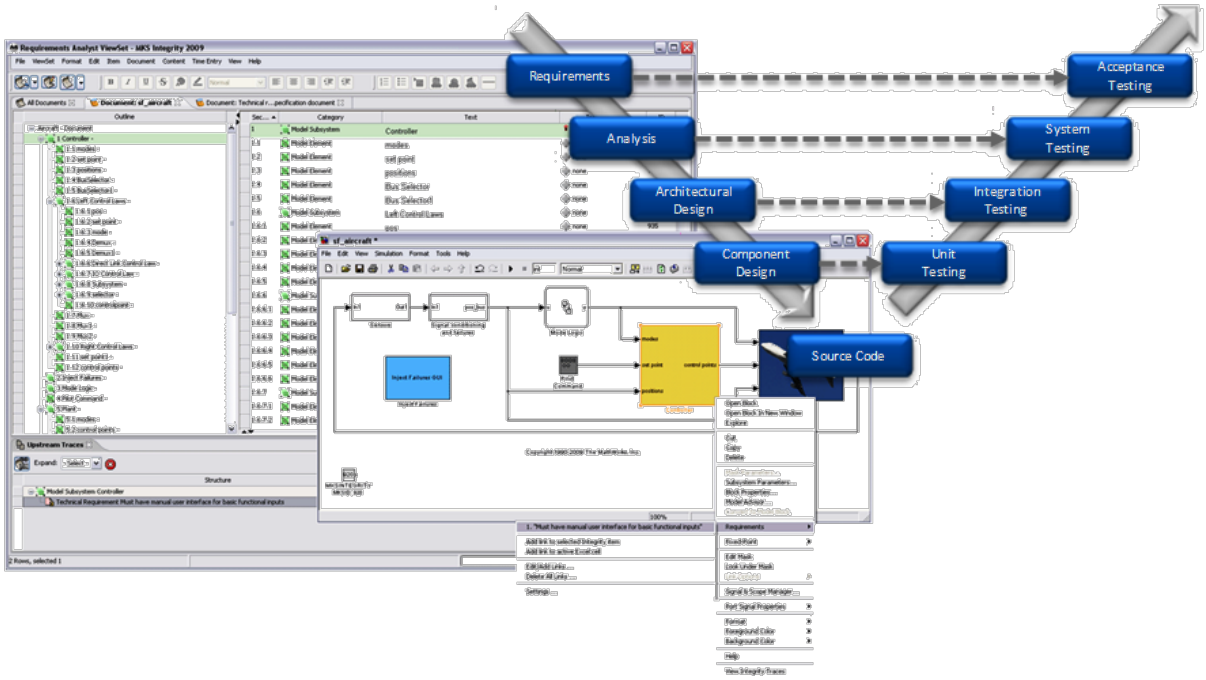
Significance

Model metrics are quantitative measures, utilized to understand model quality. This research uses metrics defined in the model-based engineering tool. [15].

The research aims to validated these metrics by correlating them to various reference entities. These references are referred to as experts opinion and failure reports. The research limited the metrics to a set of strongly correlated measures and presents them as indicators of model quality and failure probability.

Prior to presenting the calculations, this research provides an influence diagram and lists impact factors. These factors are assumed to reflect the root causes of model failure. Mapping both sets, the indications and impacts validate these assumptions and results of the aforementioned calculations.

Figure 1.1: Model-based development



1.1 Model Based Development

Model-based development is comprised of the use of charts, diagrams, and supporting visualization materials within the development process to represent and explain the processes and functionalities of the project. Examples of the appropriate environment for model-based development are Matlab Simlink and UML. The graphical or visual representations of the source codes or mathematical formulas which describe the projects are instrumental within the development process.

Model-based design supports the understanding, simulation, testing, and maintenance of the projects.[2] It also guarantees a higher quality of the embedded codes installed within the ECUs, as these codes are generated automatically by specialized tools, such as Targetlink, HDL coder, and Real Time Workshop. All of this aforementioned tools produce codes with different systemization, structure, and characteristics which are more desirable, especially within large projects.

In this decade, cars contain more than fifty ECUs. These ECUs contain between 100 to 200 millions lines of code.[1] Model-based development and automatic code generation are indispensable for the current and future of the automotive industry.

1.2 Quality Assurance, Rating and Prediction

Overall, model quality may be defined within two aspects. One, the internal quality focusing on design and development and two, the evaluation of fitness for purposes of the product.

[1] http://spectrum.ieee.org/green-tech/advanced-cars/this-car-runs-on-code

The model's internal quality is simply defined by how much a model fulfills the specifications, requirements, and guidelines. The compatibility is investigated using various tools and methods.

One of the methods is model checking, in which, models, codes and elements of libraries used in development are checked to match specifications. Other tools are used for checking compatibility with different sets of guidelines.

Model external quality is defined by how successful a model can perform in testing or in the market after final production. In other words, model quality represents how a model fails and how a failure reflects on the functionalities administered by the ECU(s). This data was collected for the set of models focused on in this study and failure reports were interpreted in order to rate these models' quality.

Meanwhile, other quality ratings and metrics for models can be obtained from a tool which was developed within *Sheible*'s thesis.[15] The key endeavor of this study consists of comparing these ratings and measurements to the failure reports of the same set of models in order to extract a list of metrics which indicate strong failure probability and then use these metrics as predictors in the quality of other models and projects.

1.3 Definitions, Metrics, Models

This section seeks to provide a foundation to support the comprehension and results of this study. In the beginning this section supplements the general objective with short of definitions of key terms that are most frequently used within this study. Then, it seeks to define and categorize metrics utilized within this study. These metrics are introduced by *Sheible* in his work regarding model quality rating.

- **Definitions**

  Model metric: A measurement of properties of portions of a model or software. The measurements are quantitative, but used and interpreted as quality metrics.

  Error: Deviation from the specification or as the actual result deviates from expected or correct result.

- **Model-based engineering Metrics**

  In *Sheible*'s thesis, an assessment tool was created to calculate eighty-four measurements of Simulink models. The tool was built using Java and utilizes (mdlx[2]) files of the models to calculate and classify measurements. Measurements are categorized based on their characteristics. Alongside measurement calculations, the Model-based engineering tool is able to rate the quality of any model in comparison to a referenced set of models (reference measurement database).

  A brief test on the quality rating functionality showed that the rating value has potential to be very sensitive to changes in the reference measurement database.

[2]Matlab model in extensible markup language(xml).

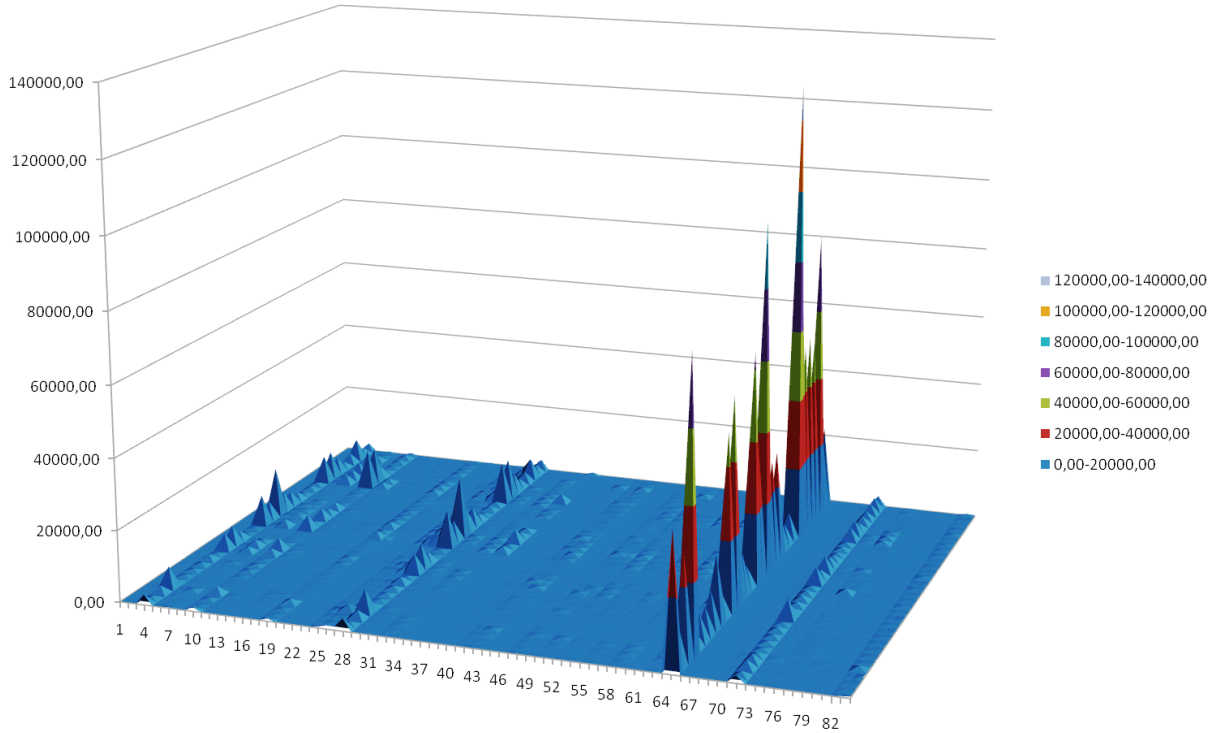
This is due to the rating not depending on a fixed reference, but rather on the relation between the measurements of the rated model and of the models included in the database. The rating is sensitive toward both the number of database models and the nature of these models.

The tool measures eighty-four metrics for every model. These metrics belong to different categories. They can be categorized according to what they represent or according to how they are calculated. Some of these metrics require extra data to be calculated or may need to be manually entered.

The illustration below (Figure 1.3) demonstrates metrics for models. Although measurements are not easily read in the illustration, the main objective is to demonstrate the great variation of metrics prior to scaling and normalization.

It is evident that the data varies largely between different models depending on different metrics. Considering the raw data without attention to size differences can threaten the validity of results due to their dependence on size differences. Therefore, normalizing all metrics and scaling them from zero to one is vital.

Figure 1.2: The metrics for all models



In order to normalize metrics, they have been divided into four subgroups. The first group consists of the metrics which depend on the number of used subsystems and for these metrics the number of subsystems is the normalization reference. This is achieved by dividing metrics of each model by the number of subsystems of the same model.(See Table 1.3) According to this, the subsystems count is one for all models, therefore we exclude this step in further calculations.

Table 1.1: Metrics normalized by # subsystems

Normalized by number of subsystems	
# arithmetic subsystems	# atomic subsystems
# clones	# cycles
# internal states	# logic subsystems
# signal routing subsystems	# testable subsystems with implicit inport data types
# testable subsystems with incoming buses	Depth of subsystem hierarchy

According to the mathematical definition provided by documentation of the MBE tool, the metrics in Table 1.2 are dependent on the blocks count in each model. The same operation was repeated, but excluding the metric number of blocks.[15]

Table 1.2: Metrics normalized by # blocks

Normalized by number of Blocks	
# bus creators	# bus selectors
# computationally expensive blocks	# data store memory
# data store memory without read access	# data store memory without write access
# data store read	# data store write
# disabled and unresolvable library links	# disabled library links
# disabled library links with modified structure	# froms
# global goto blocks	# gotos
# ground blocks	# library links
# magic constant uses	# magic constant values
# named constants	# potential data typing problems
# saturates	# stateflow blocks
# subsystem annotations	# targetlink errors
# targetlink functions	# targetlink warnings
# terminator blocks	# unit delays
# unit delays in forward direction	# unused results
# used block types	# used dd variables[1]
# used libraries	# violated misra guidelines
global complexity	# crossed lines
# line annotations	# lines

Metrics displayed in the table below (Table 1.3) are not required to be normalized, as they are independent of the model size.

Table 1.3: Averages

Averages	
øannotation length	øbus size
øgeometric signal length	øindependent paths
øinput interface width	øinstability of block
øinterconnectedness	ølocal complexity
øoutport connections	øoutput interface width

[1] dd stands for data dictionary.

øsignal length	østateflow state count
øsubsystem child count	øsubsystem screen height
øsubsystem screen width	

Metrics shown in Table 1.3 placed a zero-value for all investigated models. They have been excluded from all calculations after this step was completed. The purpose for the absence of values is that most of these measurements require external data files or need to be manually entered. None of these files were available. However, the MBE tool includes these metrics within the rating. In Chapter Four, the effect of including these metrics on the rating reliability is discussed.

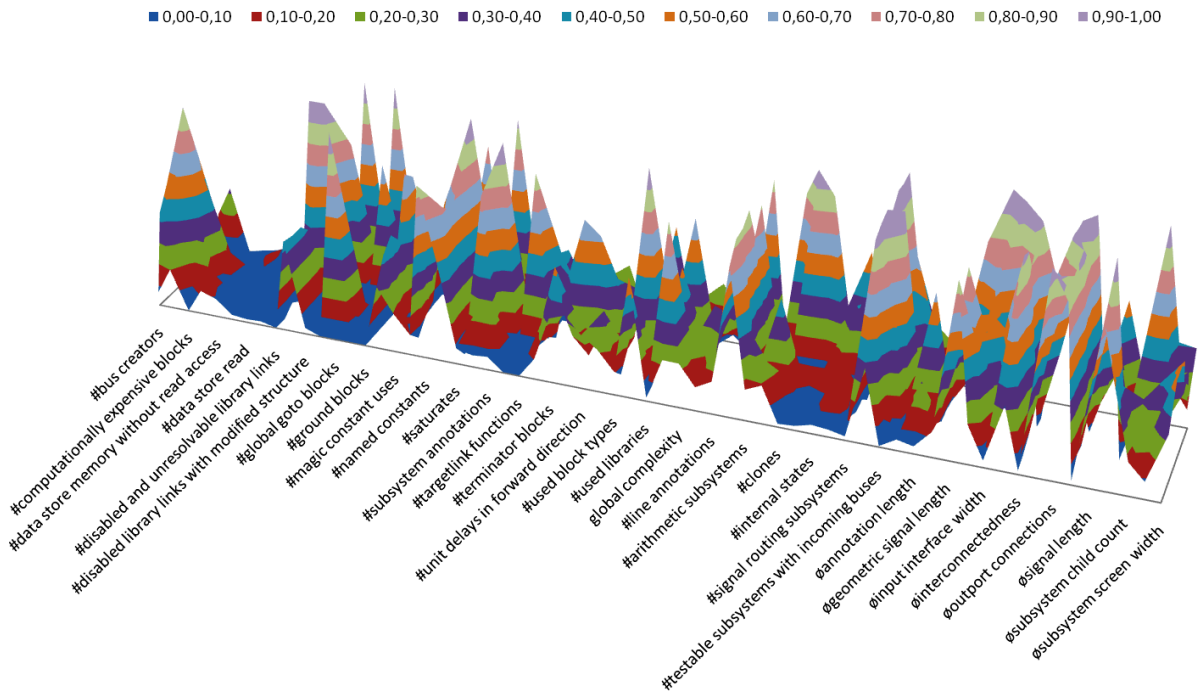
Table 1.4: Metrics which maintain zero value due to lack of data.

No value	
# violated d space guidelines	# architecture violations
# configurable subsystems	# custom code blocks
# model infos	# ports without dd entry
# range violations	# requirements
# scalings without dd entry	# switch subsystems
# targetlink block saturates	# test cases
# unreachable states	# used dd types
# used modeling patterns	Passed test cases
Requirements coverage	Test coverage
Worst case execution time	

After the normalization of metrics, values were scaled to be between zero and one by dividing the values of metrics within all models by the maximum value.

The figure below (Figure 1.4) displays how the data is reshaped in a homogeneous fashion rather than in the aforementioned, which can be seen in Figure 1.3. The shape represents the sixty-two metrics for the key eight models focused on within this research.

Figure 1.3: The sixty-two metrics for the main eight models



- **Models:** The total number of the investigated models used in this research is thirty-six. These models are a subset of one of the development repositories which contains over seventy models. The chosen models maintain a similar structure in terms of Autosar components. The MBE tool was used to calculate and find all metrics for these models. Out of the thirty-six models are the eight models used within *Sheible*'s work.

  This research focuses on these eight models, due to an extensive knowledge in comparison to the other models. The remaining models were also used to test the tool sensitivity to changes within the reference measurement database and their measurements for the principle component analysis on all metrics. For four of the models, the tool aborted operations whilst calculations for a particular metric, therefore they have been excluded.

Chapter 2

Related Work and Mathematical Foundations

The issue of failure prediction based on quality metrics is a rich topic in software quality engineering. There is a focus on object-oriented metrics. There are only a few, particular papers focusing on model metrics. This chapter aims to illustrate the most significant hypothesis which aimed to be validated by this research.

The purpose of this is to show the value and popularity of this topic, but also to display the level of difficulty in grasping strong correlations between metrics and failures. There is a wide variance of metrics and mathematical approaches to aggregate the data.

The last section of this chapter provides an overview of the most utilized metrics and average the opportunity to highlight the difference between spearman and person correlation.

2.1 Research on validation of software metrics as quality indicators

Most research on this topic follow the same three-step building sequence. A fundamental first step must be to define the metrics to be measured, the hypothesis of the relationships between the aforementioned metrics, and failure occurrence. The second step consists of constructing and presenting the mathematical model of how data will be aggregated. Additionally, metrics must be measured and targeted projects must be defined.

The third and final step is obtaining the results. It must be determined whether the hypothesis holds true or fails and the reasoning as to why. The same sequence is followed in presenting the extracted work from related papers within this study.

Main metrics to be targeted when mining the metrics of object-oriented projects are some or all of the CK metrics suite.[11, 19, 5] CK refers to *Shyam R. Chidamber* and *Chris F.Kemerer*, who in 1994 introduced six metrics for the object-oriented design.[17]

Further targeted metrics are also object-oriented metrics as couple cohesion and inheritance. Cohesion measures the strength of the relation between the functionalities of a model.[10] In general, most metrics are centered around concepts of complexity, cohesion, inheritance, and measurement of size.[22, 4, 20]

An example of the hypothesis tested by an abundance of researchers is also similar. It is almost always assumed that the increase of complexity metrics correlates with the number of post-release failure. As do the weighted methods per class, coupling between object classes from the CK suite.[19, 11]

Researchers tend to target software issues of projects developed within their respective companies. For example, Internet Explorer, Process Messaging Component, DirectX, and Net Meeting, among other projects.[11, 13, 12] Due to researchers targeting only a limited set of projects developed in their own companies within the same field, the validity of the data is threatened. This concern is further discussed within the final chapter.

Challenges for researchers exceed finding correlations and validating hypothesis. The most comprehensive challenge is to form a set of metrics which consistently correlate with the failure and conjoin the aspect of analysis with additional projects.[11]

2.2 Research on model metrics and model quality rating

Due to research on model metrics being significantly less prominent than on object-oriented metrics, supplemented studies are few and far between. One question which arises is whether findings will be different if the same metric is measured firstly on the model and secondly on the automatically generated code from the model.

In *Prabhu's* [14] workfocusing on complexity metrics, he shows that metrics correlate strongly when they are measured on the code and on the model. *Prabhu's* study regards this conclusion as a possible improvement on the quality of outsourcing within the automotive industry. This is due to the ease of software quality prediction through measuring metrics for the model prior to software development.

Sheible's work is central in terms of model quality through the tool he developed. This study and the research within it, is highly dependent on *Sheible*'s tool. Chapter Four and Chapter Five attempt to validate the tool as ratings and of metrics as quality indicators.

The illustration in Figure 2.1 is taken directly from *Sheible*'s thesis, documenting the MBE tool.[15][p 100] The illustration demonstrates how the tool handles both model data and measurements' database.

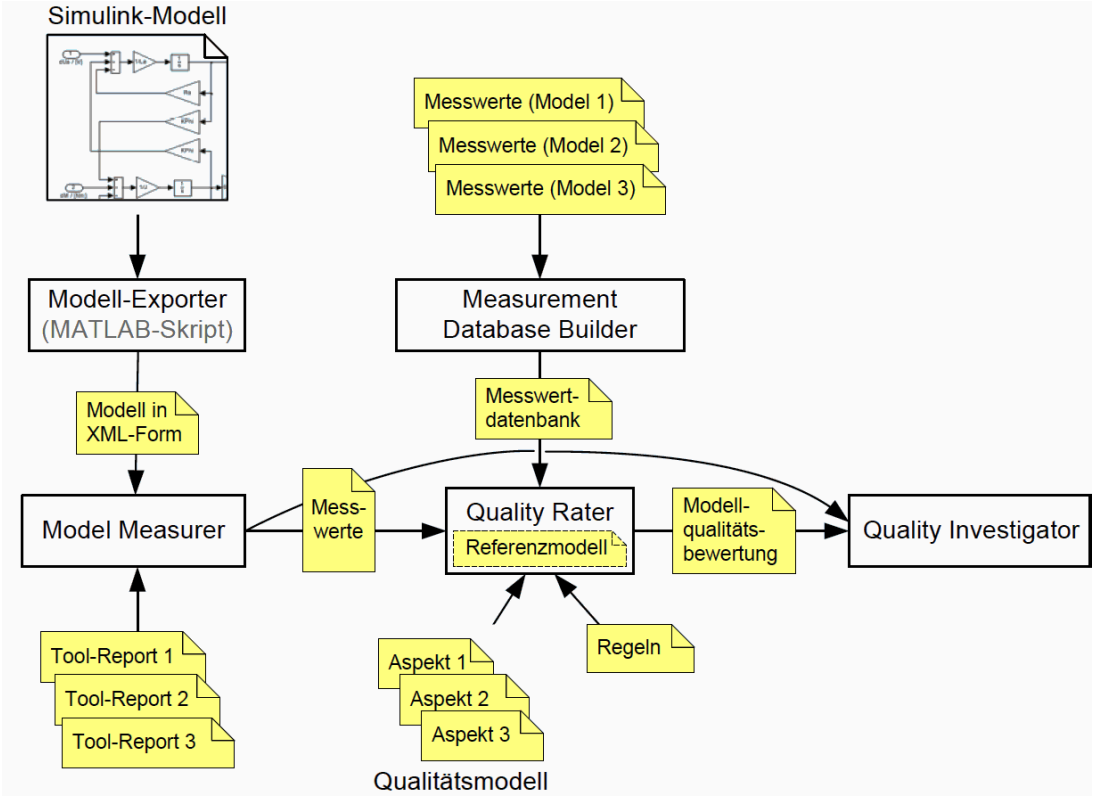


Figure 2.1: MBE tool prototype structure

2.3 Mathematical techniques of processing and aggregating software, and model statistics

Validating the metrics as quality indicators is done be calculating the correlation between the metrics of an entity and the failures reported from the same entity. Section 1.3 presented the preparation of metrics to be used in the correlation calculations.

This research adopts the spearman ranking as the main technique in finding correlations between different entities. Pearson correlation is also used when possible, but not often. One of the reasons why is due to parts of the data obtained are originally a ranking, opposed to a raw value. Additionally, the pearson correlation is more sensitive to nonlinearity than the spearman ranking. This is due to different formulas for each ranking.[9, 18]

Other methods utilized in analyzing metrics' data are related to outliers and principal component analysis, focusing on relations between metric sets and attempting to reduce dimensions and increase coherence. Additionally, in attempt to see which subsets of metrics effect and represent the rest of the data.[1, 16, 10, 11]

The spearman correlation coefficient is simply calculated by:

$$\rho = 1 - \frac{6 \sum {d_i}^2}{n(n^2 - 1)}$$

where $d_i = x_i - y_i$, x_i being the first rank and y_i being the other rank.

Chapter 3

Impacts on Model Quality

Modeling is a vital process for the development of the ECU components within the automotive industry. The increasing role of the ECU and its participation in all the operations inside the automotive system increases the sensitivity of safety, functionality, and efficiency within an automobile to the quality of the ECU and the software components they contain.

The quality of the ECU relates to how successful the unit is in executing all operations and functions it is expected to execute. Any failure of an ECU is caused by one or both of hardware or software problem. This study focuses on the quality of embedded software components implemented through Matlab Simulink. Therefore, the primary consideration of failures are from a defect in the software, more specifically within the model(s).

This chapter illustrates root causes of defects in the model. These root causes may belong to any stage or procedure of the model development and implementation. We begin by exploring the varying key parts of model development, and continue by mapping some of the metrics measured in this study to the root causes of the defect.

Mapping the measurements to the root causes and impact factors on the model quality will enable us to drive legitimate hypothesis regarding the relations between the model metrics and the amount of bugs in the final product. Utilizing the influence diagram will support the grouping and reduction of the different metrics and roots to simpler and more measurable values.

3.1 Prevention techniques

The quality of the model is effected by techniques used to prevent the occurrence of errors during production and in finding defects during testing and prototyping. This section seeks to explain a few of these techniques.

Guidelines: The objective of using guidelines is to reduce human error. Additionally, to avoid repeating any cause of defects which were previously identified. Guidelines refers to different sets of instructions, rules, and prohibitions to be considered by developers while developing models.

Violations to guidelines are expected to be strong indicators of model quality. It is assumed that a number of violations correlate positively with proneness to failure, or negatively with model quality.

Model checking: This is used to algorithmically check whether a model satisfies a specification. In general, model checking is an independent, on-going process, aiming to maintain the quality of libraries used in implementing a model.[6]

Model checking contributes to the quality of the model. It is vital that produced models match requirements and specifications. It is equally important for the structure of a model to enable checkers to reach every state, and furthermore, to check every property, ensuring compatibility.

Static Analysis: This refers to a family of techniques within program analysis, in which a program is not actually executed, but rather analyzed through tools in order to produce useful information about the program itself.

Static analysis is used to uncover bugs or potential defects, as well as basic statistics and measurements regarding the program. From these statistics, code metrics may be calculated.

Testing: This refers to the process of product execution to identify the difference between actual and expected behavior. Bugs which are found during testing are important in order to ensure enhancement of the product.

The tests' reachability, coverage, and the number of test cases are vital indicators. Discovering bugs during the testing process could provoke quality improvement if the bug is deciphered, but it can also indicate a failure of static analysis. Therefore, it is possible to find additional bugs during the testing process itself.

3.2 Changes

An additional group of factors which may influence the quality of a model are project changes. Whether the change consists of fixing bugs or of altered release changes on the model, they may create possibilities for secondary bugs.

However, fixing pre-existing bugs in a model is assumed to increase quality. The major issue is whether fixing bugs leads to changes outside the root cause of the bugs. Therefore, these changes may formulate supplementary impacts on quality.

Model Reuse: This refers to the same model used in variant modeling contexts. In other words, a distinctive domain-specific modeling language. Changes may be introduced to a model in order for it to because reusable. These changes may reproduce defects or effect the overall quality of the model.[21]

Feature Requests: This refers to changes, which might be a response to requests from users or a test group.

Fixed Bugs: This refers to changes applied on a program to fix or exclude errors.

3.3 Impacts on understanding

Product quality is dependent on a developer's understanding of requirements and users' understanding of application and usage. Many external factors may effect understanding.

Although impacts are difficult to measure, they are determinant in terms of model quality. With sufficient information from project management and after sales, they may become measurable.

Cascading of development stages: This specifically refers to how well results of each development stage passes to the next.

Product: Many factors may effect product comprehension:

- Readability: refers to naming styles and formatting consistency.
- Architecture: refers to the documentation, subsystems' mapping, concepts, and interfaces (call/write violations).
- Use of multiple languages or modeling domains: The use of multiple language and modeling domain variations may effect product comprehension.
- Requirements: refers to the amount of requirements and detail level.

Project: This refers to the number of developers working on the project, including their background and experience. This takes into account at which stage a developer joined the project and the efficiency of communication and collaboration between team members. Additionally, this factor takes into consideration if team members have been changed or replaced. Furthermore, this factor also regards the available resources, such as time, budget, etc.

3.4 Mapping and measurability of impacts on model quality

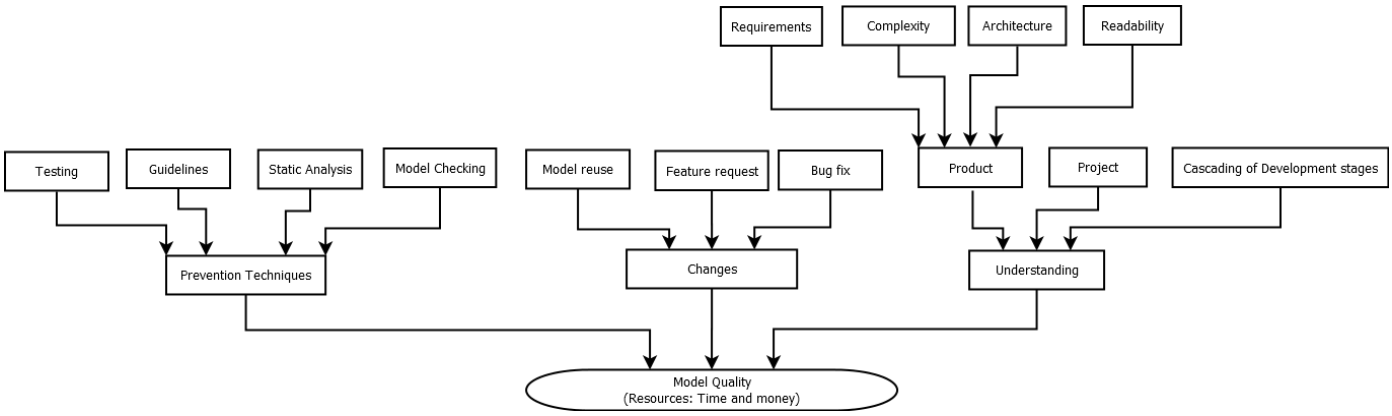
This section aims to map the most useful metrics based on experimental work to the influence diagram. By mapping these metrics, we can understand and rationalize the relation between the entities represented by these metrics and the probability of failure. Table 3.1 demonstrates the correspondence of metrics with the influence diagram.

Unfortunately, the majority of the few metrics in the Table 3.1 couldn't be calculated due to a lack of information. Based on the assumptions mapped in the influence diagram, Section 6.2 Interprets between MBE metrics and ,failure reports, and experts opinion.

Table 3.1: Mapping of sample MBE tool metrics to the influence diagram.

Section	Impact factor	Correspondent metric
Prevention techniques	Guidelines	#targetlink warnings
		#violated misra guidelines
		#violated d space guidelines
	Testing	Test coverage
		Passed test cases
		#test cases
		#unreachable states
Understanding	Architecture	#range violations
		#architecture violations
	Requirements	Requirements coverage
Changes		Instability of block

Figure 3.1: Influence Diagram



Chapter 4

Validation of MBE Tool as Quality Rater of Simulink Models

This chapter provides a validation of the use of metrics, ratings, and findings of the MBE tool, and MBE thesis. Metrics for models in this study are calculated using the MBE tool. Furthermore, this chapter demonstrates a deeper understanding and critique of the rating processes' internal steps, both methodological and mathematical.

4.1 Main functions of the MBE tool

The model quality rating tool is multi-step. The first stage consists of measuring metrics for every model. Measuring the metrics for a model requires a set of information files. The models are imported as (mdlx) files rather than (mdl) files. Additional files are required within the second stage. The rating of the model quality is the final stage.[15]. (See Figure 2.1)

The second stage involves the construction of a reference measurement database, representing a reference to which the rated model is compared and a scale to which the metrics of one model are normalized. The last stage consists of the aggregation of metrics supported with visualization utilities, enhancing the understanding of the strengths and weaknesses of the rated model.

4.2 Tests on the methods and findings of the tool

Provided sufficient data availability, the MBE tool is generally dynamic and comprehensive in terms measurement administration. Yet, the main criticisms of the tool are concerned with what follows measuring model metrics.

Some concerns are methodological, others mathematical. All concerns formulate two questions: Does the rating of the tool always correlate with the experts opinion? Does the mathematical processing of the data effect the validity of the metrics?

4.2.1 Methodological concerns

1. **Database content**

The database represents the main reference for rating and normalization. It is fundamental to understand how the database itself is built and what the its components consist of. The database, at the very least, requires the measures of three models; none of which is the actual model to be rated. There is no maximum limit on the size of the database.

From the available measures for the models included in the database, the arithmetic mean is extracted and then the measurements of every metric for all models are normalized to the mean. The output values are what the model rating depends on. The test demonstrated the change of the amount and types of models included in the database for rating a single model resulted into a remarkable shift in the final rating of the model.

Figure 4.1 displays the change of rating between 50% and 80% following the database size changing from three to thirty-one models.

Figure 4.1: The quality rating of four models changes with the database size

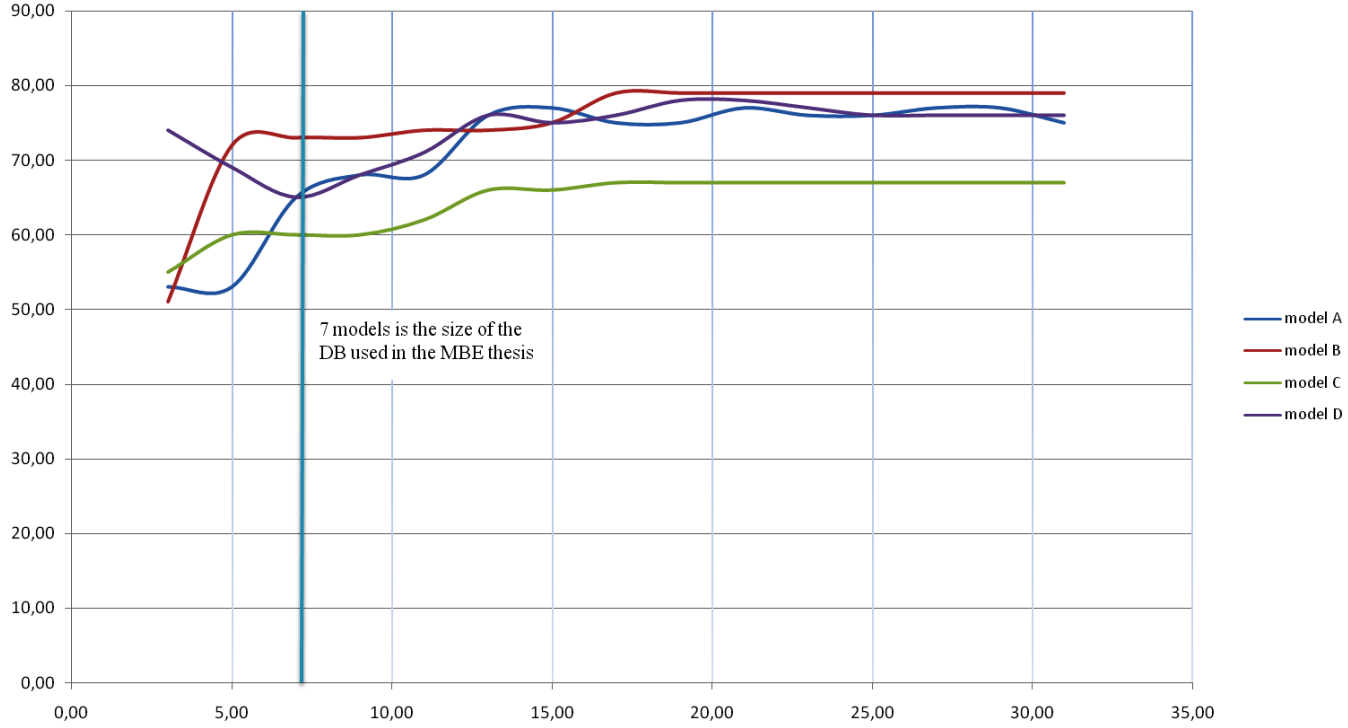
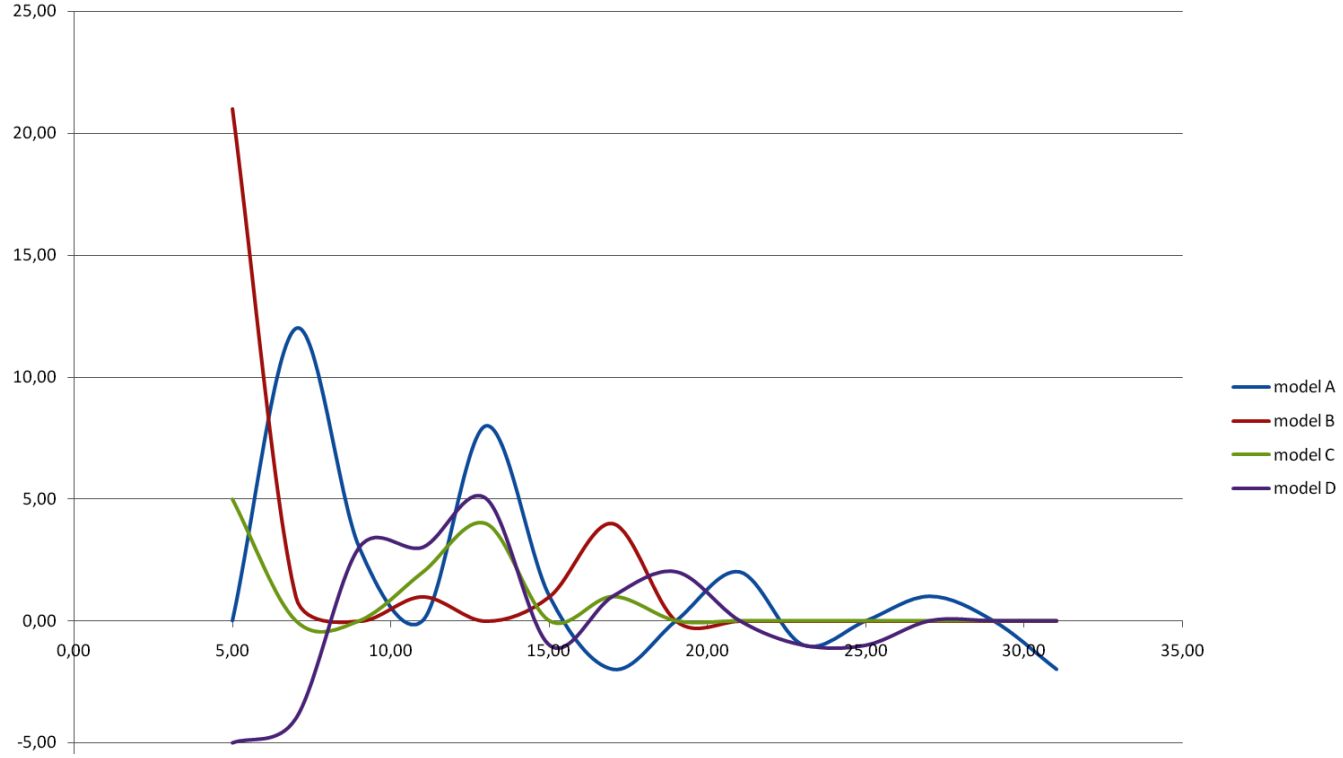


Figure 4.2 demonstrates how change decreases when database size increases. Overall, when a database size exceeds ten models, a model's quality rating shows more stability.

Figure 4.2: The change in the quality rating of four models according to the change in database size



2. Experts Opinion

Within his thesis, *Sheible* presents a remarkably high correlation between the tool ranking and the experts ranking of a selected set of models. After recalculating the rating of the same model to different versions of reference measurement databases, this finding did not hold true.

Table 4.1 and Figure 4.3 demonstrate the correlation change between experts ranking and the MBE tool ranking. The main conclusion is that these two entities do not always correlate, and furthermore, that there may be a possible database formation to be considered corrupted or having a negative effect on quality ranking. The same is implied for failure reports ranking.

The first ranking corresponds to the one presented in the MBE thesis.[15] The database consists of the same eight models focused on within this study. When a model is rated, it is compared to the other seven models.

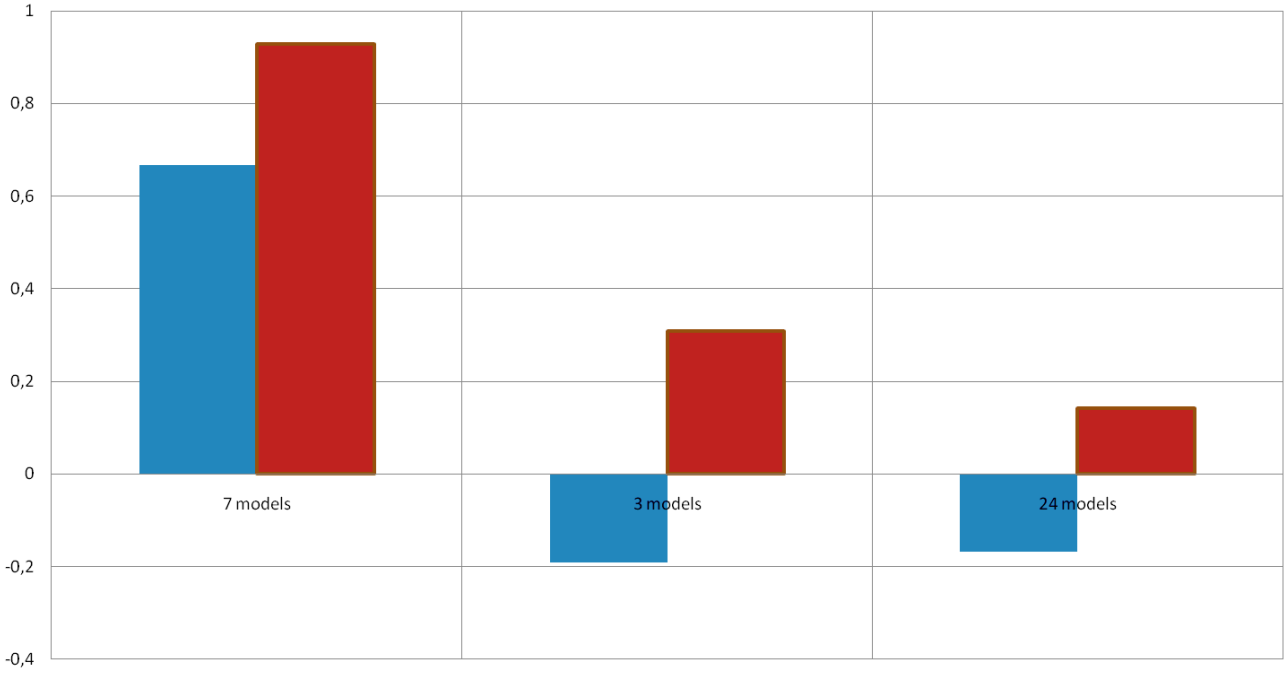
The second ranking is in comparison to a database of three models. It represents what could be deemed as the worst case scenario for a database, involving the minimum possible size.

In the final ranking, each of the eight models are rated in comparison to the additional twenty-four models.

Table 4.1: The correlation between experts ranking and MBE tool ranking

Database size	Database stability	spearman correlation
7 models	instable, dependent	0,92
3 models	stable,independent	0,31
24 models	stable, independent	0,14

Figure 4.3: The correlation between the tool and the failure reports ranking (blue,left) and the correlation between the tool and experts ranking (red,right)



4.2.2 Mathematical concerns

This section traces the processes performed on data. The objective is to uncover any steps which may effect data credibility or usability.

1. Replacing missing data in the model measurer

   The required data for at least ten metrics need to be entered manually. The tool substitutes these values with zeros when no data is provided. This procedure of compensation is not neutral. It directly effects the rating of the model by a set of false positive. These values are explained in greater detail further on within the study.

2. Metric normalization in the database builder

   Building the database is a vital step for the remainder of the tool work. The database acts as a reference to which the model is compared and rated. Therefore, it is vital to understand two pressing determinants. First, any modifications on numbers and second, why the size and contents of the database are paramount within the final rating.

In the MBE tool, the reference measurement database builder uses the global complexity as normalization factor. However, global complexity is not exactly a measurement that represent the size as the #blocks or #subsystems do. Also, according to our assumptions global complexity is an important measurement and using it for normalization may reduce its usability in the aggregation of the quality rating.

3. Metrics scaling in the quality rater

   Prior to metric aggregation in the last step of quality rating, every metric is rated individually with a value between zero and one. This value is determined by comparing a metric's original value to the reference, which in this case is the database. Several threats and concerns appear in this stage. Mainly, the concern is how to deal with a measurement outside its own boundaries and how the tool evaluates the metric value of zero when both boundaries are zero.

4. Metrics aggregation in the quality rater

   The final step of rating is the aggregation. The aggregation is performed using the formula set forth in *Sheible*'s thesis. Several questions arise at this stage: Does the multi-step aggregation lead to a different result than one-step aggregation? Does the grouping of metrics based on category and definition effect the rating result or is the result independent of grouping and categorization?[15, 8]

   The following equation is the equation used to aggregate the metrics:

   Exponent=1; Threshold = 20%

$$AggregatedEvaluation = \underbrace{\frac{\sum_i \ evaluation_i}{\#evaluation}}_{Arithmeticmean} \cdot \underbrace{\left(\frac{\#evaluationsoverThreshold}{\#evaluations}\right)^{Exponent}}_{Dampingfactor}$$

Chapter 5

Validation of MBE Metrics As Quality Indicators

The documentation of the MBE tool[15] introduces the correlation between the tool quality rating and expert opinions on eight of the models. This study investigates the correlations between three main entities: the rating extracted from failure reports, expert opinions, and the tool quality rating(s) on different reference measurement databases. The intent of this chapter is to assess the validity and coherence between the tool ranking and the ranking provided by other indicators.

The absence of correlation between these entities may be attributed to varying causes to be discussed within the last chapter of this study. None of the three entities is fully reliable. Many issues emerged while questioning the sufficiency of expert rankings and to what level failure reports from after sales form a reference in order to compare other rankings with it.

5.1 Mining the failure reports

This study collected all available reports on failures in the functionalities controlled by an ECU which contain any of the eight models mentioned above. The reports were then interpreted for purposes of conversion from their original, written description into numeric values in order to rank the models according to their 'failure score'.

The after sale department reports complications and provides a wide range of data regarding its level of importance, timing, and a written description of the problem. Taking into consideration all provided information, this study deliberates four specific entries: amount of problems reports, their occurrence, priority, and severity.[3] A demonstrative example of the failure reporting scoring system for every model may be seen in Table 5.1.

$$failurescore = \underbrace{(Occurrence)}_{1,2,3,4} . \underbrace{(Priority)}_{1,2,3,4} . \underbrace{(Severity)}_{1,2,3,...,10,11}$$

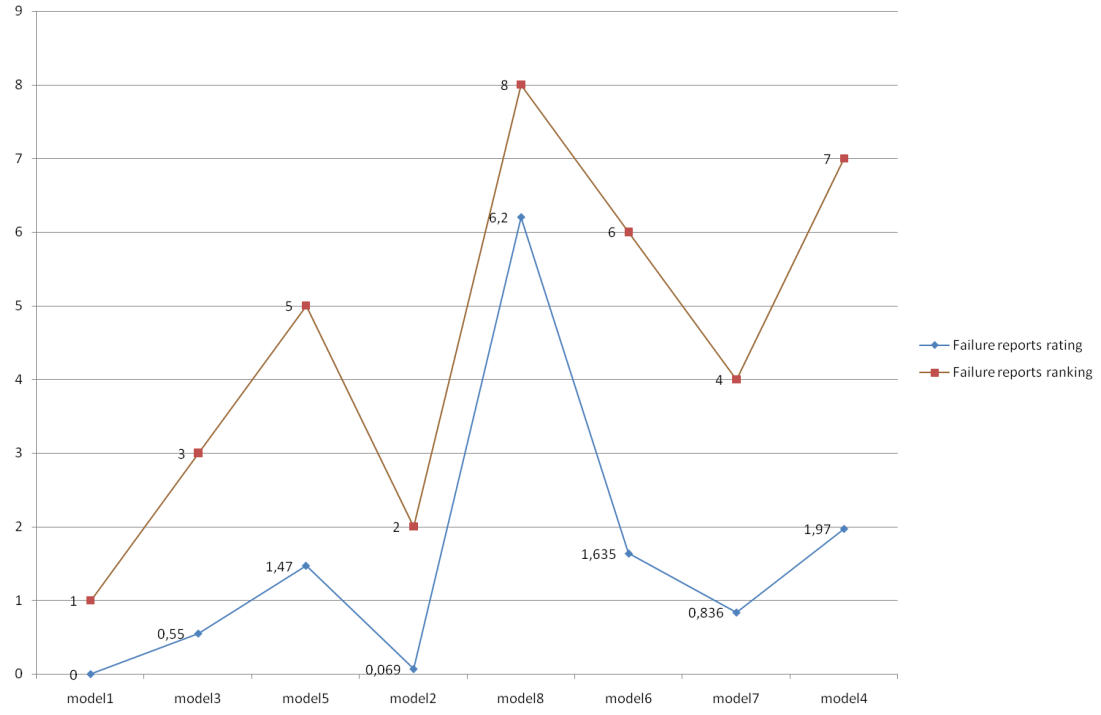
The table below (Table 5.1) is simply an example.The values within it do not refer to any specific model. The original description of the three entities was in a written format. They have been converted with consideration of avoiding zero due to the scoring formula consisting of multiplication of the three entities. Additionally, one represents no data, as it is a neutral value in multiplication.

Table 5.1: Example on the scoring of reported failure to rate the models

Model 00 Reported failure	Occurrence	Priority	Severity	Score	sum	Normalized by #Blocks
11	1	3	11	33	346	3,46
12	1	3	5	15		
13	1	3	5	15		
14	2	4	5	40		
15	4	4	5	80		
16	1	3	5	15		
17	1	4	11	44		
18	4	3	5	60		
19	4	1	11	44		

Figure 5.1 displays the rating and scoring of models focused on within this chapter.

Figure 5.1: Failure reports rating and model quality ranking



5.2 Correlations with failure reports ranking

This section aims to validate the use of failure reports ranking and experts ranking by correlating them to each other and the tool. This adopts the tool ranking based on twenty-four models, used as the reference measurement database.

Table 5.2 demonstrates the rating of models based on experts ranking and bug rating based on an interpretation of failure reports. The correlation between both rankings is '0,67'. On a reliability level for both rankings, a moderately positive correlation is indicated.[9] Additionally, this further supports both rankings as valid, comparative to a model's metrics and tool ranking.

Table 5.2: Failure reports ranking vs. experts ranking

Model[1]	Failure reports rating	Failure reports Ranking	Experts Ranking	Spearman Correlation
1	0[2]	1	2	
3	0,55	3	1	
5	1,47	5	4	
2	0,069	2	3	
8	6,2	8	7	
6	1,635	6	6	
7	0,836	4	8	
4	1,97	7	5	
				0,67

Figure 5.2: Failure reports ranking vs. experts ranking

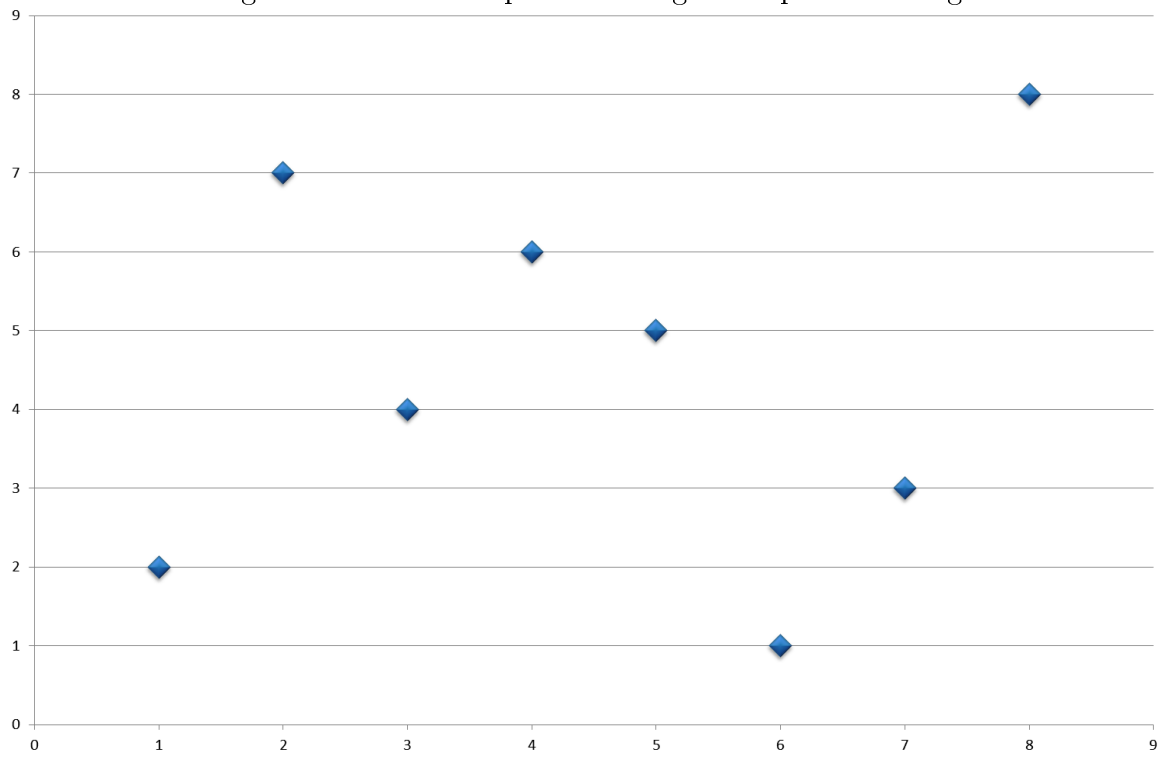
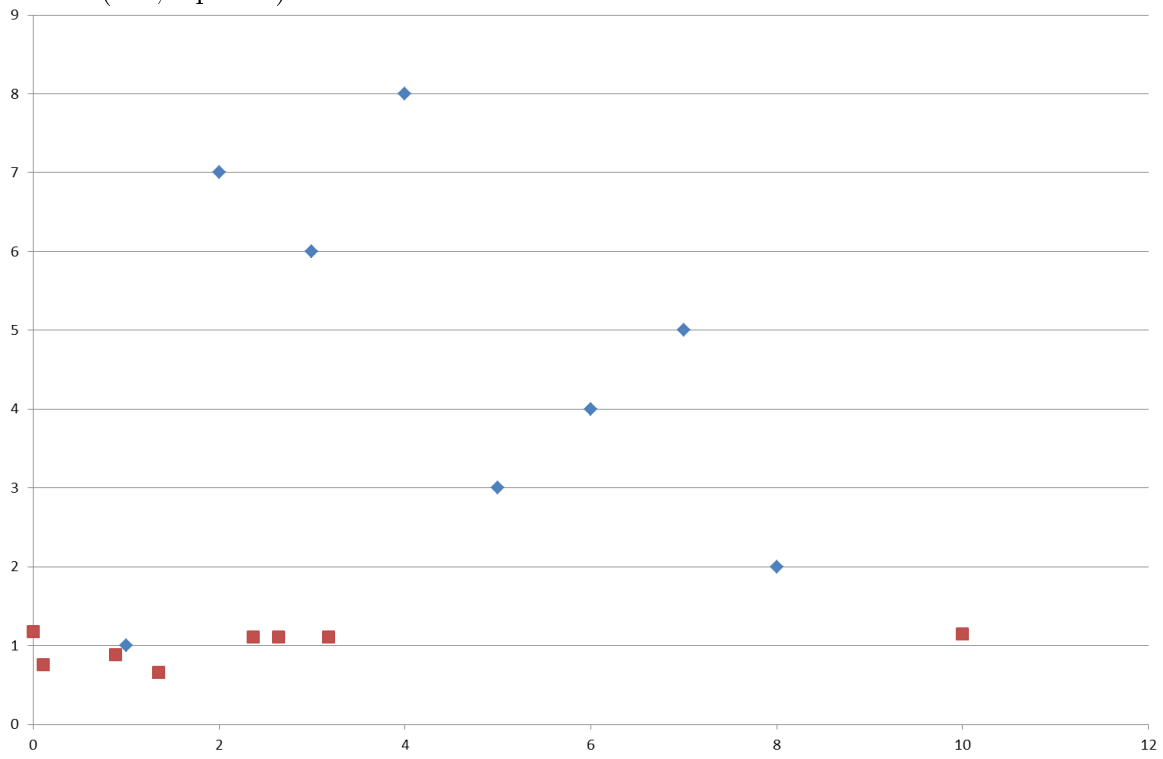


Table 5.3 and Figure 5.3 correlate the failure reports ranking with the MBE tool ranking. The table shows no significant correlation existing between the tool ranking and the failure reports ranking.

Table 5.3: Correlations between MBE tool and failure reports

Model	MBE tool[3]		Failure reports		pearson ranking	spearman ranking
	rating	ranking	rating	ranking		
1	0,73	1	0	1		
3	0,55	6	0,55	3		
5	0,69	3	1,47	5		
2	0,47	7	0,069	2		
8	0,71	2	6,2	8		
6	0,69	4	1,635	6		
7	0,41	8	0,836	4		
4	0,69	5	1,97	7	-0,02	-0,17

Figure 5.3: Correlations between MBE tool and failure reports. Spearman (blue, dots). Pearson (red, squares).



5.3 Correlations of MBE metrics and MBE tool ranking

This section presents the correlation between the MBE metrics and the MBE ranking. The objective is to understand which metrics are more influential on the final quality rating measured by the tool.

[1]Though models are listed in alphabetical order, they are display as numeric values for purposes of data protection.

[3]Through an independent database of twenty-four models.

[2]No errors in the available reports.

Figure 5.4 visualizes the aforementioned correlations, displaying metrics which gave similar rankings of the MBE tool. Table 5.4 shows the metrics that correlate moderately or strongly with the MBE tool quality ranking.

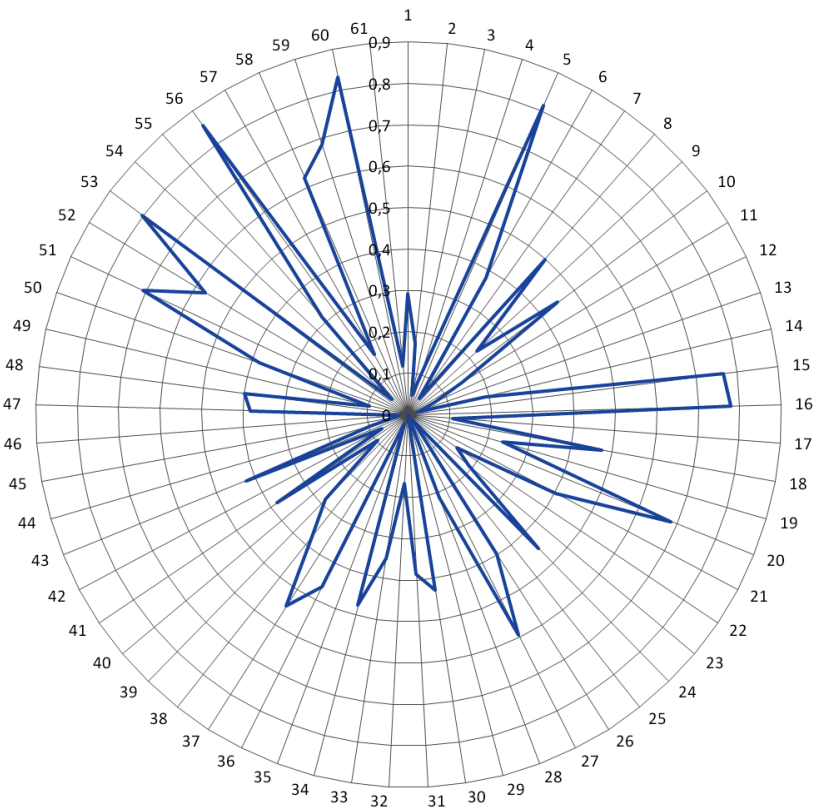
Every metric was used to rank the models. The model with the smallest metric is ranked as one (representing the optimum) and the model with the highest metric is ranked eight (representing the worst).

The metric ranking is interpreted as the following: When a metric correlates positively, it signifies the lower a metric is, the better its quality. When a metric correlates negatively, it signifies the higher a metric is, the better its quality.

Table 5.4: MBE metrics that correlate moderately or strongly with MBE tool ranking

#	MBE metric	spearman correlation	#	MBE metric	spearman correlation
56	øoutput interface width	0,85	60	øsubsystem screen height	0,83
5	#clones	0,81	53	øinterconnectedness	0,80
16	#disabled library links with modified structure	-0,77	15	#disabled library links	-0,76
51	øinput interface width	0,70	20	#internal states	0,684524
59	øsubsystem child count	0,68	58	østateflow state count	-0,62
27	#named constants	-0,59	52	øinstability of block	0,57
37	#testable subsystems with incoming buses	0,54	8	#cycles	0,5

Figure 5.4: Absolute correlations between MBE tool metrics and MBE tool quality ranking.



5.4 Correlations of MBE metrics and Failure reports

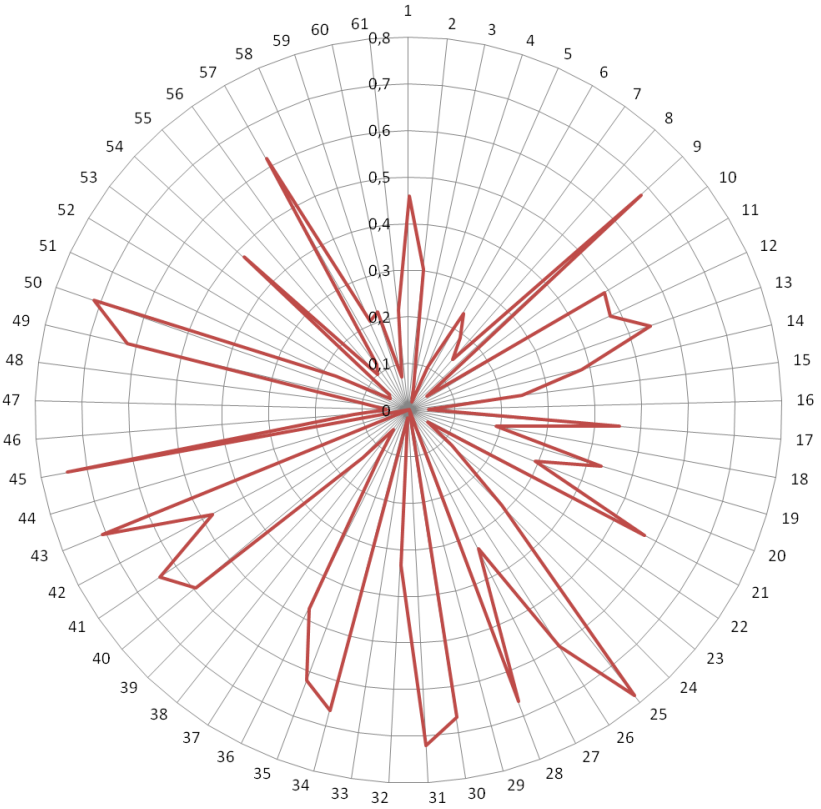
The objective of this section is to find a set of metrics correlating with failure reports ranking and utilize this set as a group of indicators on model quality and failure probability.

Table 5.5 and Figure 5.5 show correlations beginning with the metric showing the highest absolute correlation. Metrics which displayed weak correlations have been excluded. Figure 5.5 visualize the same correlation, focuses on absolute values of correlations, and excluding the signs of correlations.

Table 5.5: MBE metrics that correlate moderately or strongly with failure reports ranking

#	MBE metric	spearman correlation	#	MBE metric	spearman correlation
25	#magic constant uses	-0,77	45	depth of subsystem hierarchy	0,74
31	#stateflow blocks	0,72	50	øindependent paths	0,71
43	#used libraries	0,71	9	#data store memory	-0,67
28	#potential data typing problems	0,66	30	#signal routing subsystems	0,66
34	#targetlink warnings	0,66	41	#used block types	0,64
35	#terminator blocks	-0,62	49	øgeometric signal length	-0,62
57	øsignal length	-0,62	26	#magic constant values	0,62
40	#unused results	-0,60	21	#library links	0,57
13	#data store write	-0,54			

Figure 5.5: Absolute correlations between MBE metrics and failure reports.png



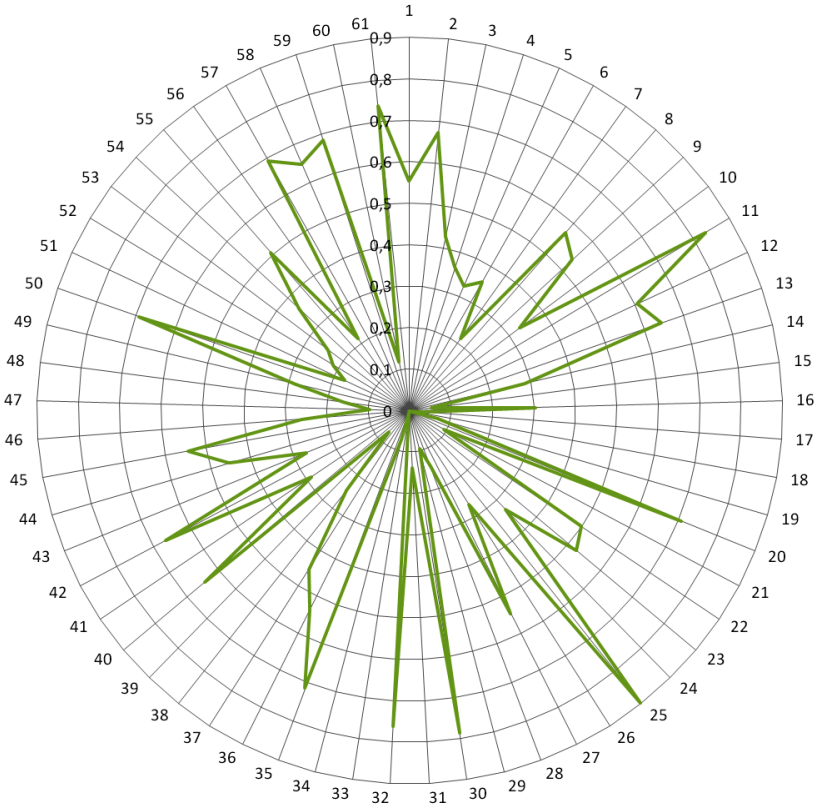
5.5 Correlations of MBE metrics and experts ranking

This section aims to find a set of metrics which indicate model quality base on correlation between metric ranking and experts ranking. Table 5.6 displays metrics which correlate moderately or strongly with experts ranking. Figure 5.6 visualize correlations in absolute values, excluding the signs of correlations.

Table 5.6: MBE metrics that correlate moderately or strongly with experts ranking

#	MBE metric	spearman correlation	#	MBE metric	spearman correlation
25	# magic constant uses	-0,89	11	# data store memory without write access	0,83
30	# signal routing subsystems	0,78	32	# subsystem annotations	-0,76
61	øsubsystem screen width	-0,73	35	# terminator blocks	-0,75
20	# internal states	0,71	50	øindependent paths	0,70
57	øsignal length	-0,70	59	øsubsystem child count	0,69
2	# atomic subsystems	-0,67	42	# used dd variables	-0,67
58	østateflow state count	-0,65	13	# data store write	-0,64
40	# unused results	-0,64	12	# data store read	-0,61
8	# cycles	0,57	1	# arithmetic subsystems	-0,56
27	# named constants	-0,55	45	ødepth of subsystem hierarchy	0,54
9	# data store memory	-0,54	36	#testable subsystems with implicit inport data types	0,53
23	# lines	-0,52			
55	øoutport connections	0,51	22	# line annotations	-0,5

Figure 5.6: Absolute correlations between MBE tool and failure reports.



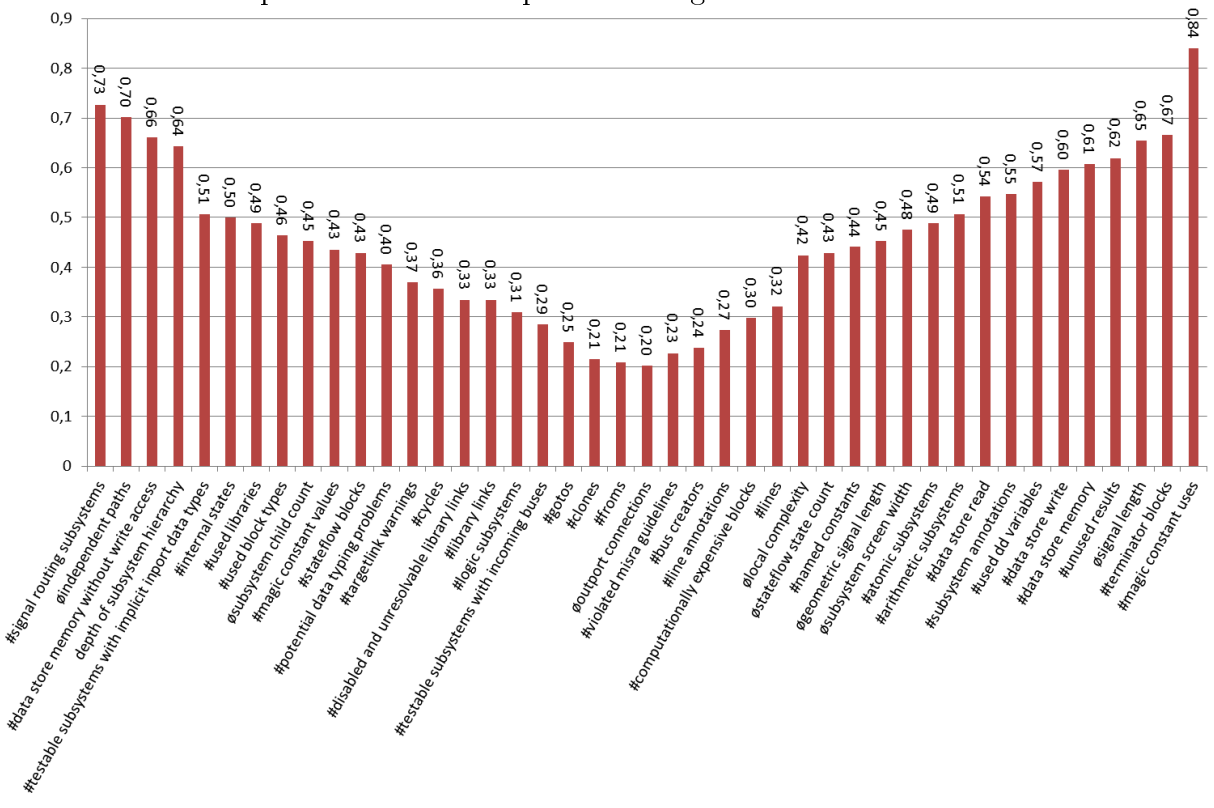
5.6 Validation of subset of MBE metrics as Quality indicators

Based on the correlations found in Section 5.4 and Section 5.5, the objective is to construct a list of metrics correlating with both entities; failure reports and experts opinion. Table 5.7 represents the calculation of the average correlation with both entities.

Table 5.7: MBE metrics which have strong or moderate øcorrelations with experts and failure reports ranking

#	MBE metric	spearman correlation	#	MBE metric	spearman correlation
25	#magic constant uses	-0,84			
30	#signal routing subsystems	0,73	50	øindependent paths	0,70
35	#terminator blocks	-0,67	11	#data storememory without write access	0,66
57	øsignal length	-0,65	45	depth of subsystem hierarchy	0,64
40	#unused results	-0,62	9	#data store memory	-0,61
13	#data store write	-0,60	42	#used dd variables	-0,57
32	#subsystem annotations	-0,55	12	#data store read	-0,54
1	#arithmetic subsystems	-0,51	36	#testable subsystems with implicit inport data types	0,51
20	#internal states	0,50	2	#atomic subsystems	-0,49

Figure 5.7: Absolute values of MBE metrics which have moderate or strong average correlation with experts and failure reports ranking.



Chapter 6

Conclusion and Discussion

The previous chapters presented the theoretical research and the mathematical analysis, which were done in order to find a set of metrics that indicate model quality. Chapter One and Chapter Two illustrated main concepts, tools, and related work, which this research is built upon. Based on related work, Chapter Three displayed the assumptions of factors and their impact on model quality.

Prior to presenting calculations within Chapter Five, Chapter Four validated the MBE tool and critically analyzes its methodology and mathematical structure. The predominately experimental portion of this research is located within Chapter Five. Chapter Five gave a detailed presentation and visualization of meaningful calculations.

This chapter summarizes and analyzes the most significant findings within this research. Furthermore, it provides a critique of data sources and highlights both weaknesses and strengths. Moreover, it discusses the prospect of utilizing these findings in the future, offering a window of improvement for results to increase robustness of data sources.

6.1 Results and Findings

This section summarizes the main findings in the same sequence of the research.

- **Influence Diagram:**

  The influence diagram maps factors which effect quality. Provided sufficient measurement, it holds the potential to become probability network. The probability network would systematize the prediction of model quality based on the factors' values and weights. However, using probability networks require much more data and needs the impact factors to be measured accurately.[7]

- **MBE tool:**

  The MBE tool is powerful in terms of the detailed metrics it provides. However, the rating functionality within the tool is extremely sensitive and requires special attention within the construction of the reference measurement database.

- **Experts ranking and failure reports ranking:**

  The experts ranking and failure reports ranking have not always correlated with the MBE tool ranking. Nevertheless, they correlated with each other, therefore, making them safe indicators for model quality..

6.2 Interpretation of validated metrics

Based on calculations within Section 5.6, metrics in Table 5.7 may be considered valid indicators of model quality. Table 6.2.1, Table 6.3, and Table 6.2.3 display these metrics and interpret their correlations.

Metrics rank models in ascending order; i.e. the best rank is for a model with the lowest metric. Accordingly, in the case of positive correlation, low metric indicates high quality while a high metric indicates low quality.

6.2.1 Metric with positive correlation

Table 6.2.1 displays the metrics that correlated positively associated with the interpretation of the result for each metric. The positive correlation implies that using the entities measured by these metrics decrease the quality of the model.

Table 6.1: Interpretation of positively correlated metrics.

#	MBE metric	spearman correlation	Interpretation
30	#signal routing subsystems	0,73	This implies using less subsystems enhance the quality. We have no primary assumption regarding this metric.However, signal routing is a kind of reusing the signal. Therefore, th
50	øindependent paths	0,70	Independent paths map conceptually with complexity. Based on this correlation less independent paths lead to better model quality
11	#data store memory without write access	0,66	This implies that limiting the access to the data store memory to the the blocks in its same subsystem enhance the quality and it corresponds positively to our assumption.

45	depth of subsystem hierarchy	0,64	this metric represents complexity. Deeper is worse in term of model quality. This support our assumption
36	#testable subsystems with implicit inport data types	0,51	More testable subsystem indicates here to worse quality which contradict our assumption about the testability increasing the quality.
20	#internal states	0,50	Internal states represents complexity. Therefore, this metric also supports the assumptions

6.2.2 Metric with negative correlation

In the case of negative correlation, low metric indicates low quality while high metric indicates high quality. Table 6.3 displays the metrics that correlated negatively associated with the interpretation of the result for each metric. The negative correlation implies that using the entities measured by these metrics increase the quality of the model.

Table 6.3: Interpretation of negatively correlated metrics.

#	MBE metric	spearman correlation	Interpretation
25	#magic constant uses	-0,84	The use of magic constants is expected to decrease the quality but this metric strongly suggests the opposite.
35	#terminator blocks	-0,67	These blocks terminate unconnected outputs. The correlation implies that more of these blocks increases the quality. however, terminator blocks and terminated outputs indicate on the use of blocks with unnecessary output or not using an existing output. This correlation contradicts our assumption.
57	øsignal length	-0,65	Signal length is a size measure. The correlation implies that high signal length indicate bitter quality.
40	#unused results	-0,62	Unused results as in terminator blocks, whether it is primely unnecessary or it should be used but it's not. Theoretically, this correlation contradict our assumptions

9	#data store memory	-0,61	More data store memory means is understood as the existence of data store memory on subsystem levels not on top level . Which positivly correspond to our assumption
13	#data store write	-0,60	This finding implies that the quality increase with the increment of this entity. This finding is not expected or mapped to any assumption.
42	#used dd variables	-0,57	This finding match our expectations. Using more data dictionary variables improves the quality.
32	#subsystem annotations	-0,55	This correlation is not mapped to any assumption or impact factor. It represents the model statistically and reflect the size of the model.
12	#data store read	-0,54	This finding implies that the quality increase with the increment of this entity. This finding is not expected or mapped to any assumption.
1	#arithmetic subsystems	-0,51	This correlation is not mapped to any assumption or impact factor. It represents the model statistically and reflect the size of the model.

6.2.3 Metrics with weak or no correlation

Out of sixty-one validated metrics seventeen metrics did not positively or negatively correlate, or weakly correlated (Less than |50%|). Table 6.2.3 displays these metrics and interpret their correlations. Generally, the value of metrics does not indicate model quality. In other words, the quality is independent from metric value.

Several metrics in the Table 6.2.3 represents key factors of impact which were assumed to effect model quality (i.e. øinstability of block and global complexity).

The absence of correlation between these metrics and the model quality can be referred to many factors. The calculation methods used in evaluating these metrics may deffer from the methods which are used in the related work that we built our assumption upon. Other reason could be the lack of the data from the source files or loss of data during normalization. The other interpretation implies our assumptions earlier are not accurate.

Other metrics in Table 6.2.3 measure simple statistics of different block types and functions (i.e. # global goto blocks and target link functions) and the graphical geometry of the model (i.e. øoutput interface width and øsubsystem screen height). may be useful if acting as a descriptive group for a model, however, they do not have the ability to indicate model quality as individual values.

Table 6.4: Metrics with weak or no correlation

#	MBE metric	spearman correlation	#	MBE metric	spearman correlation
43	#used libraries	0,49	2	#atomic subsystems	-0,49
61	øsubsystem screen width	-0,48	41	#used block types	0,46
59	øsubsystem child count	0,45	49	øgeometric signal length	-0,45
27	#named constants	-0,44	26	#magic constant values	0,43
31	#stateflow blocks	0,43	58	østateflow state count	-0,43
54	ølocal complexity	-0,42	28	#potential data typing problems	0,40
34	#targetlink warnings	0,37	8	#cycles	0,36
14	#disabled and unresolvable library links	0,33	21	#library links	0,33
23	#lines	-0,32	24	#logic subsystems	0,31
6	#computationally expensive blocks	-0,30	37	#testable subsystems with incoming buses	0,29
22	#line annotations	-0,27	19	#gotos	0,25
44	#violated misra guidelines	-0,23	5	#clones	0,21
17	#froms	0,21	55	øoutport connections	0,20
4	#bus selectors	-0,17	56	øoutput interface width	0,17
38	#unit delays	-0,15	10	#data store memory without read access	0,14
16	#disabled library links with modified structure	-0,13	60	øsubsystem screen height	-0,10
53	øinterconnectedness	0,10	18	#global goto blocks	0,10
15	#disabled library links	0,10	52	øinstability of block	0,08
46	global complexity	-0,07	29	#saturates	0,05
48	øbus size	0,04	39	#unit delays in forward direction	0,04
47	øannotation length	-0,02	7	#crossed lines	-0,01
33	#targetlink functions	-0,01	51	øinput interface width	0,00

The validated assumption in this section states that the metrics with moderate and strong correlations may indicate model quality and aggregate to the rate of model quality. Additionally, these models may be integrated into a guideline utilizing these findings in order to enhance the model development process.

6.3 Quality of data sources and their threats to validity

This section highlights key threats for both main validity and data source quality used within the research.

- **Targeted models:**

  Due to the amount of provided information, this research focuses on eight models. The findings have potential to be more robust and reliable if a larger validation group was available. Also, the reference database will have more reliability with larger amount of models used to build the reference reference measurement database.

- **Model measurements:**

  The quality of the MBE tool is high considering the metrics. The main concern is due to many metrics requiring external data files that were unattainable. The exclusion of these metrics resulted in a loss of data, which may have effected results. Although assumptions regarding the impact of these metrics cannot be validated due to their inability of being calculated.

- **Failure reports:**

  Although failure reports are considered a reasonable reference, ranking based on them can be more robust provided additional reports and further details within them. In the next section (Section 6.4) the prospect of improving these reports is discussed.

- **Experts ranking:**

  This ranking could vary based on the background of inquired experts. This entity is subject of human error.

6.4 Future work

This section discusses the prospect of improvement within this research.

- **MBE tool:**

  A set of guidelines may be followed to enhance the tool's performance.

  1. Database size: The database should contain more than ten models in order to deliver a more stable rating.

  2. Database content: The database should not include any of the evaluated models when a set of models are rated in conjunction.

  3. Manual entry of metrics: Metrics that are not automatically calculated should be entered manually to protect the rating from compensation applied by the tool.

- **Resource Expansion:**

  A major potential in improving research results is to inquire more data on all possible levels, targeting additional models to increase reliability, measuring additional metrics to increase usability, asking more experts' opinions to create a stronger base for comparisons, and finally, to collect more failure reports and uniform the time period in which failures were reported.

- **Rating the model quality based on correlated metrics:**

  The list of correlated metrics could be aggregated to rate the quality of the model. Mainly, this subset of metrics is an alternative to the complete set of metrics used in

the quality rating. Also, the correlations between the MBE metrics and the general MBE tool rating presented in Section 5.3 could be utilized to support this expansion.

- **Test the findings for different models and different projects:**

  In order to make the findings of this research more reliable, the findings can be tested against different models and projects. If the same metrics correlate similarly regardless of the model or the project, then these findings can be generalized .

- **Principal Component Analysis and Probability Network:**

  Further analysis of data may be done by interpreting metrics using PCA. Additionally, using a probability network to map metrics as numerical values and predict quality based on these maps.[7]

  The optimum scenario for the development of this work includes mapping a metric set to the impact factors over an influence diagram. By using enough data, every impact factor is assigned to a value which represents its importance or influence. While metric values are provided via the model, applying both sets of values to a probability network may lead to a measured, accurate prediction of model quality.

Summary

According to the research findings, there are a set of metrics which correlate with both entities; failure reports and experts ranking. A portion of this set respond positively to assumptions provided in the influence diagram. Another portion oppose these assumptions regarding their effect. Within the former section, it was found that for these findings to be generalized, they need to be validated and tested against broader groups of models and projects.

Furthermore, the former section highlighted potential for integrating indicators (validated metrics) and impact factors (influence diagram) into one map, or network of probabilities, in order to predict failures and measure quality based on an accurate evaluation of impacts and indicators.

Bibliography

[1] Principal component, canonical correlation, and exploratory factor analysis. In Neil H. Timm, G. Casella, S. Fienberg, and I. Olkin, editors, *Applied Multivariate Analysis*, Springer Texts in Statistics, pages 445–514. Springer New York, 2002.

[2] Franz Huber Jan Philipps Bernhard Schaetz, Alexander Pretschner. Model-based development. Institute Of Informatics, Technical University Munich.

[3] Bernd Bertsche and Bernd Bertsche. *FMEA:Failure Mode and Effects Analysis*. VDI-Buch. Springer Berlin Heidelberg, 2008.

[4] S. COUNSELL and T. HALL. A theoretical and empirical analysis of three slice-based metrics for cohesion. School of Information Systems, Computing and Mathematics Brunel University, Uxbridge, Middlesex. UB8 3PH, UK.

[5] Garima Verma Dr. M.P. Thapaliyal. Software defects and object oriented metrics . an empirical analysis. Analysis.International Journal of Computer Applications.

[6] A.P. Sisla E.M. Clarke, E.A. Emerson. Automatic verification of finite state concurrent systems using temporal logic. ACM Trans. on Programming Languages and Systems.

[7] Ronald A. Howard and James E. Matheson. Influence diagrams. September 2005.

[8] Hartmut Pohlheim Jan Scheible. Automated model quality rating of embedded systems. Daimler AG - Group Research and Advanced Engineering, Model Engineering Solutions GmbH.

[9] MD Stuart G. Silverman MD Kelly H. Zou, PhD Kemal Tuncali. Statistical concepts series:correlation and simple linear regression. Department of Radiology, Brigham and Women's Hospital (K.H.Z., K.T., S.G.S.) and Department of Health Care Policy (K.H.Z.), Harvard Medical School.

[10] John W. Daly Lionel C. Briand, Jurgen Wust and D. Victor Porter. Exploring the relationships between design measures and software quality in object-oriented systems. Fraunhofer Institute for Experimental Software Engineering Kaiserslautern, Germanyl.

[11] Andreas Zeller Nachiappan Nagappan, Thomas Ball. Mining metrics to predict component failures. Microsoft.

[12] Brendan Murphy. Nachiappan Nagappan, Thomas Ball. Using historical in-process

and product metrics for early estimation of software failures. Microsoft Research.

[13] Thomas Ball Nachiappan Nagappan. Using software dependencies and churn metrics to predict field failures: An empirical case study. First International Symposium on Empirical Software Engineering and Measurement.

[14] Jeevan Prabhu. Complexity analysis of simulink models to improve the quality of outsourcing in an automotive company.

[15] Jan Scheible. Automatisierte qualitaewtsbewertung am beispiel von matlab simulink-modellen in der automobil-domaene. Phd Thesis,Tubingen University. To appear in 2012.

[16] Jonathon Shlens. A tutorial on principal component analysis. Center for Neural Science, New York University New York City, NY.

[17] Chris F.Kemerer Shyam R. Chidamber. A metric suite for object oriented design.

[18] Lorentz JANTSCHI Sorana-Daniela BOLBOACA1. Pearson versus spearman, kendall's tau correlation analysis on structure-activity relationships of biologic active compounds.

[19] Ramanath Subramanyam and M.S. Krishnan. Empirical analysis of ck metrics for object-oriented design complexity:implications for software defects. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING.

[20] Rudolf Ferenc Tibor Gyimoethy and Istvaan Siket. Empirical validation of object-oriented metrics on open source software for fault prediction. OCTOBER 2005. Prediction.IEEE TRANSACTIONSON SOFTWARE ENGINEERING.

[21] Miklos Maroti Akos Ledeczi Tihamer Levendovszky, Gabor Karsai. Model reuse with metamodel-based transformations. Institute for Software Integrated Systems Vanderbilt University,Peabody ,Nashville.

[22] Lionel Briand Victor R. Basili and Walcelio L. Melo. A validation of objectoriented design metrics as quality indicators. Technical Report, Univ. of Maryland, Dep. of Computer Science, College Park, MD, 20742 USA.