Simulink Models Are Also Software: Modularity Assessment

Yanja Dajsuren, Mark G. J. van den Brand, Alexander Serebrenik, Serguei Roubtsov
Eindhoven University of Technology
5612 AZ Eindhoven, The Netherlands
{y.dajsuren | m.g.j.v.d.brand | a.serebrenik | s.roubtsov}@tue.nl

ABSTRACT

In automotive industry, more and more complex electronics and software systems are being developed to enable the innovation and to decrease costs. Besides the complex multimedia, comfort, and safety systems of conventional vehicles, automotive companies are required to develop more and more complex engine, aftertreatment, and energy management systems for their (hybrid) electric vehicles to reduce fuel consumption and harmful emissions. MATLAB/Simulink is one of the most popular graphical modeling languages and a simulation tool for validating and testing control software systems. Due to the increasing complexity and size of Simulink models of automotive software systems, it has become a necessity to maintain the Simulink models.

In this paper, we defined metrics for assessing the modularity of Simulink models. A Java tool developed to measure the defined metrics on Simulink models interfaces with a visualization tool to facilitate the maintenance tasks of the Simulink models. The modularity metrics is furthermore validated in two phases. In the first phase, the modularity measurement is validated against the experts evaluation of a system. In the second phase, we studied the relationship between metric values and number of faults. We have observed that high coupling metric values frequently correspond to number of faults. Modularity metrics will be extended to architectural quality metrics for automotive systems.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics

General Terms

Measurement

Keywords

automotive architectural quality; Simulink model; quality metrics; modularity

QoSA'13, June 17–21, 2013, Vancouver, BC, Canada.

1. INTRODUCTION

Automotive companies face strict fuel consumption demands from the market and emission limits from legislations. Particularly, CO_2 emission reduction is considered as the biggest challenge for the automotive industry for the years to come [16]. This requirement necessitates major innovations, particularly in powertrain efficiency. The powertrain of an automotive vehicle is a set of components (e.g. the engine, transmission, drive shafts, differentials, and the drive wheels) that generates power and delivers it to the road surface. Increasing efficiency of the powertrain calls for development of new and more efficient *energy managers*, software components determining the optimal use of the available power resources [49]. The fact that energy management, functionality so crucial for the modern vehicles, is delegated to software is an indication of the immanence of software in the automotive world. Indeed, since the introduction of software in vehicles thirty years ago, the amount of software has grown exponentially and nowadays is responsible for 50-70% of the total development costs of the electronics and software systems[10]. Furthermore, given that the lifetime of a vehicle is more than two or three decades [10], addition of the new functionality or fixing defects necessitate maintainable and evolvable software.

Modularity is one of the key maintainability characteristics of the ISO/IEC SQuaRe quality standard [1]. According to this standard, modularity is defined as a degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components [1]. For automotive software modularity is recognized as being paramount [38] as changing or reusing non-modular software is very costly. Automotive software is commonly being developed using model-based design tools like Simulink and Stateflow[1] together with automatic code generation tools. Assessing quality of Simulink models has become more important for automotive manufacturers due to the increasing complexity of the models and stricter safety-related requirements [25]. Large automotive Simulink models can consist of up to 15,000 building blocks, 700 subsystems and 16 hierarchical levels [47]. Current quality assessment techniques such as the Mathworks Automotive Advisory Board (MAAB) guidelines and Model Advisor from Mathworks focus mainly on configuration settings and guideline conformance rather than model quality [25]. Although there are plethora of source code quality analysis tools available, methods for assessing quality of Simulink models are still limited.

[1] `http://www.mathworks.com/`

In this paper, we study modularity of Simulink models by refining the Simulink metamodel and introducing modularity metrics. Furthermore, to facilitate application of the approach by industry practitioners, we suggest visualisation of Simulink structure, dependency, and quality metrics using the SQuAVisiT tool [48].

The remainder of the paper is organized as follows. Section 2 introduces the modularity metrics for Simulink models. Section 3 discusses the two phases of modularity metrics evaluation. Section 4 elaborates the interfacing between our modularity tool and SQuAVisiT toolset to facilitate the maintenance task. Related work is addressed in Section 5. The final section concludes the paper and lists further steps of our research.

2. SIMULINK MODULARITY METRICS

Simulink is a visual modeling language and tool for developing, simulating and analyzing multidomain dynamic systems.[2] The metamodel of Simulink is presented in Figure 1.

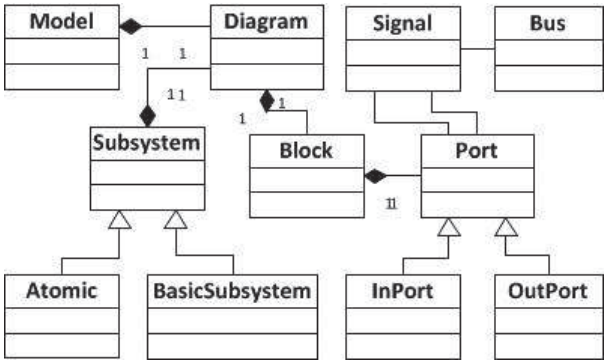

Figure 1: Simulink metamodel (revised from [8]).

A Simulink model contains a Simulink diagram, which consists of different kinds of blocks (e.g. *Transmission* model in Figure 2). *Subsystems* are blocks that contain again a Simulink diagram (e.g. *TransmissionRatio* subsystem marked by the circle). The subsystem concept enables hierarchical modeling, i.e., subsystems can contain other subsystems. We defined a special kind of subsystem as a *BasicSubsystem*, if it does not contain other subsystems (e.g. *TransmissionRatio* is also a basic subsystem). Another special kind of subsystem is an *Atomic Subsystem*, which is a subsystem that can contain a contract [8], which means blocks within an atomic subsystem are grouped together in the execution order. *Blocks*, basic elements of a Simulink diagram, communicate via input (*InPort*) and output (*OutPort*) ports: e.g. *Tin* is an input port and *Tout* is an output port of *TransmissionRatio*. A block can be connected to another block by a *Signal* via its ports. For the sake of diagram readability signals are frequently grouped in buses.

To define Simulink modularity metrics we have followed the Goal-Question-Metrics (GQM) approach [6]. Our goal was to measure modularity of Simulink models, and the general question we asked was "What modularity aspects hinder quality of Simulink models as perceived by practitioners?". Posing this question to practitioners could, however, bias their answers as they might have been tempted to give answers that might be perceived as desirable by the researcher. Therefore, we have analysed general quality reviews of third-party Simulink models carried out by control system architects and engineers of an automotive company. For every subsystem architects and engineers identify architectural problems and provide modeling comments. Thirty out of 97 problems in the review report are directly or indirectly related to modularity. Table 1 lists the modularity-relevant quality issues identified by the experts, as well as the corresponding metrics.

[2] http://www.mathworks.com/products/simulink/

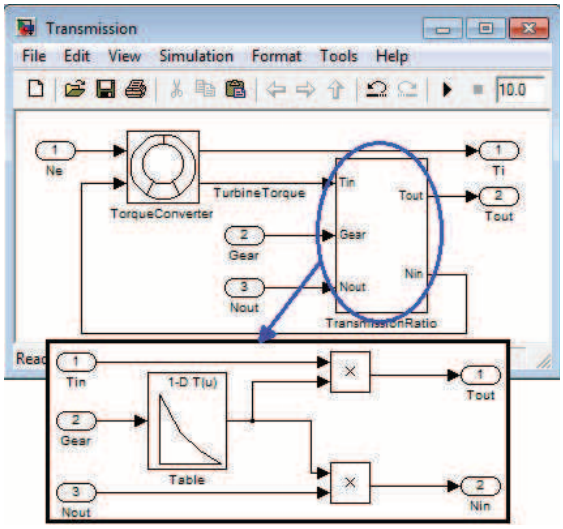

Figure 2: Simulink example model.

Issue	Derived metrics
Too many hierarchical levels of subsystems	Depth of a subsystem
Extensive use of the input bus with large number of signals	Number of input bus signals
Subsystems (blocks) that should be combined into larger subsystems	Coupling between subsystems
Similar or duplicated functionality	Considered as a future work

Table 1: Modularity-related quality review issues and the corresponding metrics.

Next we elaborate on the metrics identified in Table 1:

- Depth of a subsystem (DoS) is the maximum level the subsystem has till its basic subsystems.
- Number of input bus signals is the difference between the number of input signals (NiS) and the number of input ports (NiP). Indeed, since every input port should have a corresponding input signal and every input signal can have but one corresponding input port, the difference between the number of input signals and the number of input ports corresponds to the number of input signals that have to enter a port, which is already associated with another signal. This situation is possible only if an input bus is used. For the sake

Coupling metrics	
CBS	Coupling Between Subsystems
DSC	Degree of Subsystem Coupling
NiP	Number of input Ports
NoP	Number of output Ports
NiS	Number of input Signals
NoS	Number of output Signals
Cohesion metrics	
DoS	Depth of a Subsystem
NCS	Number of Contained Subsystems
NBS	Number of Basic Subsystems

Table 2: Modularity metrics for Simulink model.

of symmetry, we also measure the number of output signals (NoS) and the number of output ports (NoP).

- To measure coupling between subsystems (CBS) 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. CBS is close in spirit to Coupling Between Object classes [12], an object-oriented metrics referring to total number of methods of a class, which uses methods or instance variables of another class.

- In addition to CBS, we define Degree of Subsystem Coupling (DSC) to give additional weight to the output which imply more complexity: $DSC = W_i N_{is} + W_o N_{os}$, where $W_i = 1$, is weight of input dependencies, $W_o = 2$, is weight of output dependencies.

Furthermore, building on the long-standing tradition of modularity research in software engineering [36, 12, 43], we have decided to augment the metrics list in Table 1 with those reflecting common software engineering guidelines such as the number of connections between subsystems should be low (low coupling), system should contain similar or related functionalities (high cohesion) and communication between subsystems should be limited (narrow interfaces). While coupling and interface granularity have already been explicitly addressed by the metrics in Table 1, to evaluate cohesion we include the Number of Contained Subsystems (NCS) at all hierarchical levels and the Number of Basic Subsystems (NBS). In Table 2, the summary of modularity metrics for Simulink models is provided. The interface granularity metrics are integrated into the coupling metrics.

3. METRICS TOOL AND EVALUATION

We developed a Java tool to measure the defined metrics on Simulink models, since to our knowledge there is no other tool available. The tool uses a Java parser for Simulink MDL files of the ConQAT open-source tool[3], which is an integrated toolkit for continuously monitoring quality characteristics of software systems. Our tool reads Simulink MDL files with the standard structural format and generates the metrics files with the list of subsystems and the respective modularity metrics. In the following sub-sections, we discuss the results of the evaluations that were carried out in two main phases.

[3]`http://www.conqat.org/`

3.1 Expert evaluation

The first part of the validation effort was based on the expert evaluation. To this end we have randomly selected a number subsystems of an industrial application and got them evaluated by the domain experts using a scale of 1 to 10, 1 meaning worst and 10 meaning best modularity. We have opted for the 1–10 scale rather than more customary five- or seven-point Likert scales, since the 1–10 scale is the one used in Dutch schools and universities, and, hence, is familiar to the domain experts. The experts also provide the reasoning for the scores they give to the subsystems.

Results of the expert evaluation are summarized in Table 3 (left). For confidentiality reasons we abbreviate the names of the subsystems. Similarly, for privacy reasons we do not disclose the names of the experts.

Subsystem	*Experts*	
	ABC	DEF
EH	3 3 7	5 5 7
ED	7 9 7	8 6 8
SM	7 7 5	3 5 8
IDA	9 8 7	7 7 8
GS	7 6 7	7 6 8
TP	9 8 7	8 8 8
TS	3 7 1	7 7 7
BTL	9 8 7	8 8 7
CC	7 8 1	8 7 6
TSCA	9 8 7	8 8 8
TRC	7 8 5	7 8 3

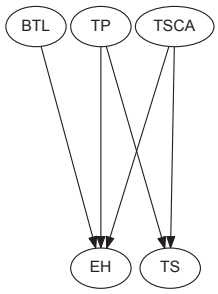

Table 3: Expert review of selected subsystems of the industrial application and the corresponding $\widetilde{\mathbf{T}}$-graph. Components missing from the $\widetilde{\mathbf{T}}$-graph are incomparable.

We start by comparing the evaluation of different subsystems. Traditionally, comparison of multiple groups follows a two-step approach: first, a global null hypothesis is tested, and then multiple comparisons are used to test sub-hypotheses pertaining to each pair of groups. The first step is commonly carried out by means of analysis of variance (ANOVA) [21] or its non-parametric counterpart, the Kruskal-Wallis one-way analysis of variance by ranks [22]. The second step uses pairwise t-tests or their nonparametric counterparts, Wilcoxon-Mann-Whitney tests [53], with Bonferroni correction [15, 46]. Unfortunately, the global test null hypothesis may be rejected while none of the sub-hypotheses are rejected, or vice versa [20]. Moreover, simulation studies suggest that the Wilcoxon-Mann-Whitney test is not robust to unequal population variances, especially in the unequal sample size case [54]. Therefore, one-step approaches are required: such an approach should produce confidence intervals which always lead to the same test decisions as the multiple comparisons. We have used the recently proposed multiple contrast test procedure $\widetilde{\mathbf{T}}$ [27] in combination with a $\widetilde{\mathbf{T}}$-graph [50]. Using the $\widetilde{\mathbf{T}}$-procedure for the "all pairs" (Tukey-type) contrast and 95% confidence level and inspecting the corresponding $\widetilde{\mathbf{T}}$-graph (Table 3 right) we can conclude that the experts prefer BTL, TP and TSCA over EH and TP and TSCA over TS. The $\widetilde{\mathbf{T}}$-procedure does not reveal consistent differences between the expert evaluations, i.e., one cannot argue that one of the experts consistently

gives higher/lower rankings than another one. Therefore, we do not exclude any of the evaluations.

We observe, however, that for individual subsystems the expert ratings are not always consistent with each other: e.g. EH is ranked 3 by experts A and B and 7 by experts C and F. To get better insights in the reasons for this discrepancy, we have discussed it with the experts. Discussion revealed that experts A, B and C interpreted modularity as coupling, while experts D, E and F interpreted modularity in terms of cohesion. Thus as shown in Table 3, we grouped the experts evaluation by coupling (A to C) and cohesion (D to F).

Before determining the relation between the coupling metrics and expert evaluation, we carried out the Kendall's τ correlation test [17] on the coupling-related metrics and identified DSC as the statistically independent metric for measuring coupling attribute. In Figure 3, a scatter plot of DSC values and an average experts' evaluation based on coupling (Exp) shows that there is a negative correlation between DSC and the expert evaluation. We calculated the Kendall's τ correlation coefficient to determine the strength of the correlation and it is $r = -0.641$, which suggests a strong negative correlation. Therefore, high values of DSC indicate poor modularity according to experts evaluation based on coupling. The subsystems BTL, TP and TSCA have low DSC values and EH and TS have high DSC values. In the review comments of the domain experts, the list of subsystems with high coupling metrics values (high values of DSC) indeed identified as subsystems that are difficult to maintain due to big dependencies. There are, however, subsystems with the purpose of input processing thus score high on coupling. Therefore, it is not necessary to take the metrics as absolute indicators of poor modularity but rather an facilitator of the maintenance process.

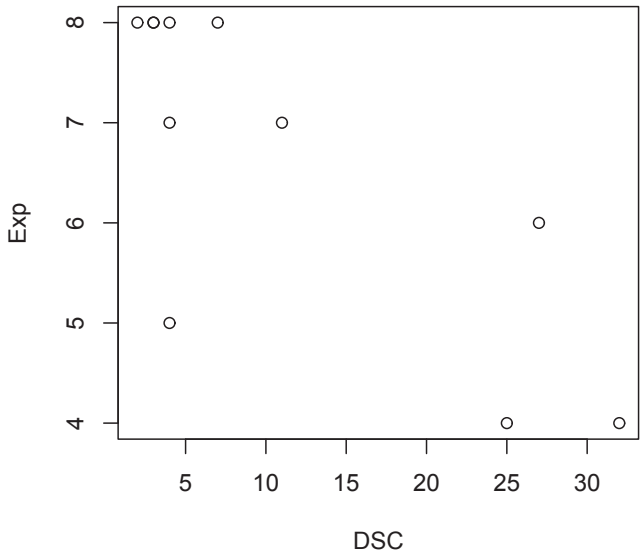

Figure 3: Relation between expert evaluation and coupling measurement.

We made the Kendall's τ correlation analysis on the cohesion-related metrics to identify the statistically independent cohesion metrics. According to the analysis, NCS and NBS have positive correlation, $r = 0.952$. Therefore, we exclude the NBS metric. Regarding the correlation between NCS, DoS and expert evaluation, there is no correlation between NCS and expert evaluation ($r = -0.238$). However, the Kendall's τ correlation coefficient, $r = -0.279$, suggests a very weak negative correlation between DoS and expert evaluation – subsystems with lower depth of a subsystem values are evaluated good by the cohesion-related experts. In the review comments of the domain experts, the list of subsystems with high hierarchal level (more than 4 hierarchal depths) identified as subsystems that are difficult to understand.

	DSC	NCS	DoS	NF
DSC	1.000	0.597	0.569	0.549
NCS		1.000	0.907	0.683
DoS			1.000	0.705
NF				1.000

Table 4: Kendall's τ correlation analysis.

3.2 Metrics evaluation

In this evaluation phase, we carried out a correlation analysis to detect if there is a relation between modularity metrics and the number of faults. A fault is an incorrect program step, process, or data definition in a computer program [18]. In this paper, we use the term fault, which exclude incorrect step and process, but refers to an error in modeling or logic that cause the system to malfunction or to produce incorrect results. It is important to measure model defects to keep control over the maintenance [7]. As stated earlier our main goal was to identify modularity aspects that hinder quality of Simulink models. By obtaining fault information and analysing the relation of faults to modularity metrics, we aim to determine quality and furthermore predict a fault-proneness or a presence of faults.

We used the data collected from the second industrial application consisting of 41 subsystems, from which 20 subsystems contain faults. Since there are a number of tied values and establishing whether any modularity metric and the number of faults are statistically dependent rather than measuring the degree of the relationship between linear related variables, we use the Kendall's τ correlation test as used in Section 3.1. Table 4 shows the correlations between the selected metrics and number of faults (NF).

The correlation coefficient infers the strength and direction of the correlation meaning a positive correlation coefficient indicates a positive relation between the metrics and defects and a negative correlation coefficient indicates a negative relation. The significance points to a probability for a coincidence. We accept a common significance level of 0.05. As presented in Table 4, the all the metrics are positively correlated with the number of faults. Based on this preliminary analysis, we can conclude that the high value of higher hierarchal levels may imply a more fault-prone system. This is in line with the studies from other programming paradigms, which show that there is a strong correlation between software maintenance effort and software metrics in the procedural and object-oriented paradigms e.g. [29] [41].

4. VISUALIZATION TOOL

As the preliminary analyses concluded that the degree of a subsystem coupling (DSC), number of contained subsystems (NCS), and depth of a subsystem (DoS) are key to assess

modularity of Simulink models, it is important to visualize these metrics in an efficient manner. For this purpose, we selected SQuAVisiT (Software Quality Assessment and Visualization Toolset) [48] as it is a flexible tool for visual software analytics of multiple programming languages. The SQuAVisiT is intended to support a software development team (including developers and maintainers) for carrying out quality assurance and maintenance tasks in an efficient way.

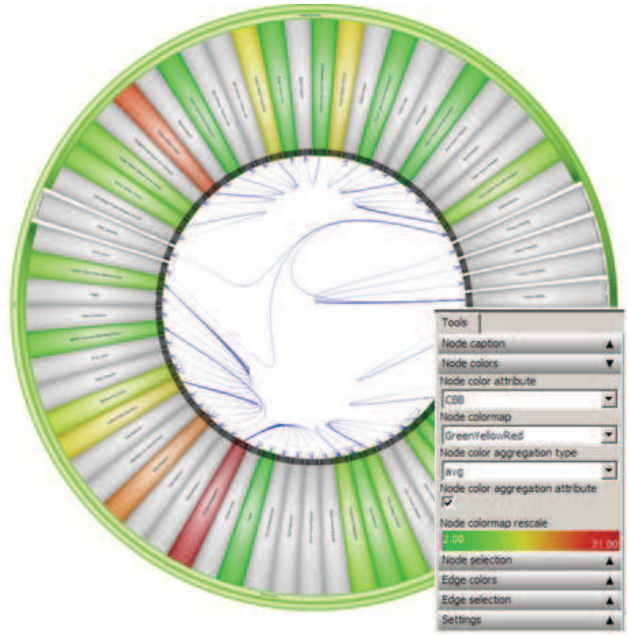

Figure 4: SQuAVisiT extended quality radial view of SolidSX tool[5]

In Figure 4, it visualizes the modularity and dependency of the industrial application that we studied. (The subsystem names here and elsewhere are blurred up due to confidentiality reasons.) The radial view was first introduced by Holten [23] and extended in SolidSX [40] and the SQuAVisiT toolset [42]. Subsystems of a Simulink model are illustrated as nested rectangles in the outer rings of the radial view. The relations between (basic) subsystems, such as input and output signals, are shown as curved arrows in blue color. The tool allows for zooming into the subsystems for more detailed views. The colors on subsystems are used to visualize values of modularity metrics. The green subsystems show the modular and red ones show subsystems which require attention for improving their modularity.

The modularity metrics are supported in the SQuAVisiT tool. Figure 4 demonstrates the visualization of software with CBS (Coupling Between Subsystems) metric, where CBS is selected in the right panel and CBS values for the subsystems are shown in numbers and respective coloring. The grey rectangles represent Bus Selectors (forking bus signals) and Bus Creators (merging signals into a bus signal). Figure 4 highlights not only subsystems with high coupling, but also shows subsystems which they depend upon, so that the developer can easily navigate and zoom into the dependencies of a particular outlier. This facilitates the signal tracing activity in Simulink and decreases the analysis time to detect coupled subsystems.

[5]http://www.solidsourceit.com/

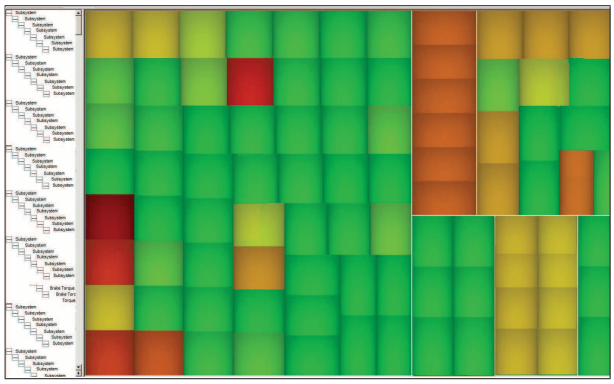

Figure 5: SQuAVisiT quality treemap view.

An example treemap view of our industrial application as illustrated in Figure 5, it shows its structural nesting highlighted with modularity measurements (green — modular, red — nonmodular). The advantage of this view is all the subsystems and their sub-components are shown simultaneously and it is easy for the architect to identify easily the subsystems, which may require further inspection [40].

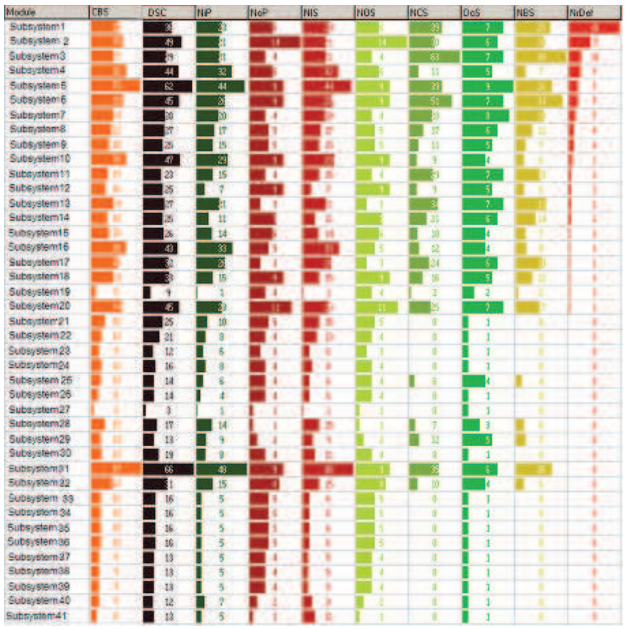

Figure 6: SQuAVisiT quality table view.

In Figure 6, we illustrate all the metric values of the subsystems including the excluded metrics (NiP, NoP, NIS, NOS, and NBS). The table contains cells as pixel bars scaled and colored by metric values as first introduced by Rao et al. [39]. The first column displays the list of subsystems and the rest of the columns list all the metric values and last column lists the number of faults of the subsystems. Subsystems are sorted by descending number of fault value. This can help the domain experts locate easily the most problematic subsystems and their respective metric values.

5. RELATED WORK

A quality model based on ISO/IEC 9126 standard for assessing internal quality of Simulink models is introduced by W. Hu et al. [25]. Six quality sub-characteristics like analysability, changeability, stability, testability, understandability, and adaptability are selected for the quality model together with respective metrics. However, modularity sub-characteristic and respective metrics are not explicitly addressed by this quality model. A metric suite to identify the most complex and instable parts of the system is introduced by Menkhaus and Andrich [32]. It measures McCabe cyclomatic complexity, instability of blocks inspired by Martin's afferent and efferent connections between blocks (CBB) (based on the interaction of blocks with other blocks), and instability of system accounting influences of the complete system on a block. Although afferent and efferent connections between blocks are used in the metrics of instability of blocks, it is not directly related to modularity. The objective of this metric suite is to guide analysis team during the risk assessment of failure modes rather than providing insight into improving modularity of the system or subsystem.

The Mathworks provide quality related tools like Modeling Metric Tool [2] [24] and *sldiagnostics* [3] to quantitatively measure the content of a Simulink model (Stateflow model as well) to improve the productivity and quality of model development (e.g. model size, complexity, and defect densities). Quality analysis metrics for measuring instability, abstractness, and complexity of Simulink models are introduced by Olszewska (Pląska) in [35] [34]. However, the modularity metrics are not explicitly addressed by the MathWorks tools and Olszewska's metrics. Modularity metrics as part of software architecture metrics are introduced by Ahrens et al. [4]. However, validation of the metrics is not provided.

Existing methods for measuring modularity are mostly intended for imperative and object-oriented (OO) software e.g. Li and Henry's OO metrics that predict maintainability (i.e. the number of methods invocation s in a class's implementation, the number of abstract data types used in the measured class and defined in another class of the system) [29], Martin's OO design quality metrics [31, 44], Chidamber and Kemerer's metrics suite for OO design (i.e. weighted methods per class, depth of inheritance, number of children, coupling between objects, response for a class, lack of cohesion of methods) [12], Lorenz and Kidd's OO software metrics [30], and design quality metrics of OO software systems of Abreu et al. [9].

In our work we have proposed a series of modularity metrics for Simulink subsystems. Since subsystems can be composed into larger ones, rather than evaluating the larger ones directly, one could have inferred the metrics values of the larger subsystems by aggregating the corresponding metric values of the subsystems composing them. This approach would be related to the metrics aggregation problem as known in software maintenance. While the most common aggregation technique, the mean, represents a central tendency and as such is unreliable for heavily skewed distributions, typical for software metrics [51], recently applied to metrics aggregation econometric inequality indices [52, 45] and threshold-based approaches [33] provide a promising way to address this challenge. More profound study of metrics aggregation for Simulink models is considered as a future work.

6. CONCLUSION AND FUTURE WORK

Due to the increasing complexity and size of Simulink models [19], it is important to assess its quality. In this paper, we focused on the modularity aspects of Simulink models and defined modularity metrics for Simulink models following the Goal-Question-Metrics (GQM) approach. We defined a modularity metrics suit consisting of 9 coupling and cohesion-related metrics and validated it with experts' evaluation and preliminary statistical analysis. We identified three independent metrics (degree of subsystem coupling, number of contained subsystems, and depth of a subsystem) based on the statistical analysis and identified a correlation between modularity metrics and number of errors. We developed a tool to measure modularity of the Simulink models and visualized the quality aspects with SQuAVisiT toolset.

Although we did a preliminary analysis on defining a relation between modularity metrics and number of faults, it needs to be explored further if the moderate correlation is satisfactory. This investigation can be compared to source code or other graphical modeling languages. The benefit of visualizing using SQuAVisit needs to be investigated further. As identified in our previous research that connectors (signals or dependencies) are not modeled as first-class objects in automotive architecture description languages (ADLs) [13], it holds for MATLAB/Simulink modeling as well. Therefore, we will refine the modularity metrics related to connectors e.g. number of subsystems using a signal, unused signals of a bus. The increasing complexity of Simulink models requires a need of higher abstraction levels. Therefore, we believe that the metrics we defined here can be applied or extended to architectural quality metrics for automotive systems [14]. In addition, the modularity metrics should take into account the increasing number of subsystems and dependencies when the system evolves. This issue has been identified in the analysis of high-cohesion/low-coupling metrics [5].

Modularity is one of the new quality sub-characteristics of ISO/IEC SQuaRE quality standard and it is related to other quality (sub-)characteristics e.g. *reusability*, *modifiability*, and *stability*. In the expert evaluation, understandability is considered as one of the key related quality characteristic as well due to the Simulink visual modeling. Therefore, we plan to extend the proposed modularity metrics with the related quality (sub-)characteristics for Simulink models. Once the quality metrics for the Simulink models for the automotive domain is thoroughly validated, it can be modified or extended further for other embedded domains.

As stated in Section 5, we also consider as future work investigation of metrics aggregation techniques for Simulink models. Finally, while in the current paper we have focussed on one Simulink model, we intend to study changes in modularity metrics during the system evolution, and investigate the applicability of Lehman's general software evolution laws [28] to evolution of Simulink models (cf. [11, 26]). Should model repositories become available, we can augment the evolutionary studies by repository mining techniques [37].

7. ACKNOWLEDGEMENT

The first three authors are supported by the Dutch Automotive Innovation Programme (HTAS) project (HTASI10002).

8. REFERENCES

[1] ISO/IEC 25010:2011 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. http://www.iso.org/.

[2] The modeling metric tool. http://www.mathworks.com/matlabcentral/fileexchange/5574.

[3] The sldiagnostics tool. http://www.mathworks.nl/help/simulink/slref/sldiagnostics.html.

[4] D. Ahrens, A. Frey, A. Pfeiffer, and T. Bertram. Designing reusable and scalable software architectures for automotive embedded systems in driver assistance. Technical report, 2010.

[5] N. Anquetil and J. Laval. Legacy software restructuring: Analyzing a concrete case. In *CSMR*, pages 279–286, 2011.

[6] V. R. Basili. Software modeling and measurement: the goal/question/metric paradigm. Technical report, College Park, MD, USA, 1992.

[7] B. Boehm. *Characteristics of Software Quality*. TRW software series / Systems Engineering and Integration Division, TRW Systems Group. 1973.

[8] P. Boström, R. Grönblom, T. Huotari, and J. Wiik. An approach to contract-based verification of Simulink models. Technical report, 2010.

[9] F. Brito e Abreu, Miguel Goulão, and R. Esteves. Toward the design quality evaluation of object-oriented software systems. In *Proc. of International Conference of Software Quality (QSIC)*, pages 44–57, 1995.

[10] M. Broy. Challenges in automotive software engineering. In *Proc. of the International Conference on Software Engineering (ICSE)*, pages 33–42, New York, NY, USA, 2006. ACM.

[11] J. Businge, A. Serebrenik, and M. G. J. van den Brand. An empirical study of the evolution of Eclipse third-party plug-ins. In *Proc. of EVOL/IWPSE*, pages 63–72, 2010.

[12] S. Chidamber and C. Kemerer. A metrics suite for object oriented design. *Software Engineering, IEEE Transactions on*, 20(6):476 –493, 1994.

[13] Y. Dajsuren, M. G. J. van den Brand, and A. Serebrenik. Evolution mechanisms of automotive architecture description languages. In *BENEVOL*, pages 24–25, 2011.

[14] Y. Dajsuren, M. G. J. van den Brand, A. Serebrenik, and R. Huisman. Automotive ADLs: a study on enforcing consistency through multiple architectural levels. In *Proc. of Quality of Software Architectures (QoSA)*, pages 71–80. ACM, 2012.

[15] O. J. Dunn. Multiple comparisons among means. *Journal of American Statistical Association*, 56:52–64, 1961.

[16] European Commission. Commission Regulation (EC) No 692/2008 of 18 July 2008 implementing and amending Regulation (EC) No 715/2007 of the European Parliament and of the Council on type-approval of motor vehicles with respect to emissions from light passenger and commercial vehicles (Euro 5 and Euro 6) and on access to vehicle repair and maintenance information, 2008.

[17] A. Field. *Discovering Statistics Using SPSS*. SAGE Publications, 2005.

[18] W. A. Florac. Software Quality Measurement: A Framework for Counting Problems and Defects. Technical report, Software Engineering Institute, 1992.

[19] J. Friedman. MATLAB/Simulink for automotive systems design. In *Proc. of Design, Automation and Test in Europe (DATE)*, volume 1, pages 1 –2, 2006.

[20] K. R. Gabriel. Simultaneous test procedures—some theory of multiple comparisons. *The Annals Mathematical Statistics*, 40(1):224–250, 1969.

[21] R. Harris. *ANOVA: An Analysis of Variance Primer*. F.E. Peacock Publishers, 1994.

[22] M. Holander and D. A. Wolfe. *Nonparametric statistical methods*. Wiley, 1973.

[23] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[24] A. Hosagrahara and P. Smith. Measuring productivity and quality in model-based design. Technical report, 2005.

[25] W. Hu, T. Loeffler, and J. Wegener. Quality model based on ISO/IEC 9126 for internal quality of MATLAB/Simulink/Stateflow models. pages 325 –330, 2012.

[26] A. Israeli and D. G. Feitelson. The linux kernel as a case study in software evolution. *Journal of Systems and Software*, 83(3):485–501, 2010.

[27] F. Konietschke, L. A. Hothorn, and E. Brunner. Rank-based multiple test procedures and simultaneous confidence intervals. *Electronic Journal of Statistics*, 6:738–759, 2012.

[28] M. M. Lehman and L. A. Belady, editors. *Program evolution: processes of software change*. Academic Press Professional, Inc., San Diego, CA, USA, 1985.

[29] W. Li and S. Henry. Object-oriented metrics that predict maintainability. *Journal of Systems and Software*, 23(2):111 – 122, 1993.

[30] M. Lorenz and J. Kidd. *Object-oriented software metrics: a practical guide*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.

[31] R. Martin. OO Design Quality Metrics - An Analysis of Dependencies. In *Workshop Pragmatic and Theoretical Directions in Object-Oriented Software Metrics*, 1994.

[32] G. Menkhaus and B. Andrich. Metric suite directing the failure mode analysis of embedded software systems. In *ICEIS (3)*, pages 266–273, 2005.

[33] K. Mordal, N. Anquetil, J. Laval, A. Serebrenik, B. Vasilescu, and S. Ducasse. Software quality metrics aggregation in industry. *Journal of Software: Evolution and Process*, 2012.

[34] M. Olszewska (Pląska). Simulink-specific design quality metrics. Technical report, Turku, Finland, 2011.

[35] M. Olszewska (Pląska), M. Huova, M. Waldén, K. Sere, and M. Linjama. Quality analysis of simulink models. In I. Schieferdecker and S. Goericke, editors, *Software Quality Engineering. Proceedings of the*

CONQUEST 2009., pages 223–240. dpunkt.verlag GmbH, 2009.

[36] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Commun. ACM*, 15(12):1053–1058, 1972.

[37] W. Poncin, A. Serebrenik, and M. G. J. van den Brand. Process mining software repositories. In T. Mens, Y. Kanellopoulos, and A. Winter, editors, *CSMR*, pages 5–14. IEEE Computer Society, 2011.

[38] A. Pretschner, M. Broy, I. H. Kruger, and T. Stauner. Software engineering for automotive systems: A roadmap. In *Future of Software Engineering*, pages 55–71, Washington, DC, USA, 2007. IEEE Computer Society.

[39] R. Rao and S. K. Card. The table lens: merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. In *Conference Companion on Human Factors in Computing Systems*, CHI '94, pages 222–, New York, NY, USA, 1994. ACM.

[40] D. Reniers, L. Voinea, and A. Telea. Visual exploration of program structure, dependencies and metrics with solidsx. In *VISSOFT*, pages 1–4, 2011.

[41] H. Rombach. A controlled expeniment on the impact of software structure on maintainability. *Software Engineering, IEEE Transactions on*, SE-13(3):344 – 354, march 1987.

[42] S. Roubtsov, A. Telea, and D. Holten. Squavisit: A software quality assessment and visualisation toolset. In *Proceedings of the Seventh IEEE International Working Conference on Source Code Analysis and Manipulation*, SCAM '07, pages 155–156, Washington, DC, USA, 2007. IEEE Computer Society.

[43] C. N. Sant'Anna. *On the Modularity of Aspect-Oriented Design: A Concern-Driven Measurement Approach*. PhD thesis, Pontifical Catholic University of Rio de Janeiro, 2008.

[44] A. Serebrenik, S. A. Roubtsov, and M. G. J. van den Brand. Dn-based architecture assessment of Java open source software systems. In *ICPC*, pages 198–207. IEEE Computer Society, 2009.

[45] A. Serebrenik and M. G. J. van den Brand. Theil index for aggregation of software metrics values. In *ICSM*, pages 1–9, 2010.

[46] D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall, 4 edition, 2007.

[47] I. Stürmer and H. Pohlheim. Model quality assessment in practice: How to measure and assess the quality of software models during the embedded software development process. In *Embedded Real Time Software and Systems*, 2012.

[48] M. G. J. van den Brand, S. Roubtsov, and A. Serebrenik. SQuAVisiT: A flexible tool for visual software analytics. In *Proceedings of the European Conference on Software Maintenance and Reengineering*, pages 331–332, Washington, DC, USA, 2009. IEEE Computer Society.

[49] V. van Reeven, T. Hofman, R. Huisman, and M. Steinbuch. Extending energy management in hybrid electric vehicles with explicit control of gear shifting and start-stop. In *American Control Conference (ACC)*, pages 521–526, 2012.

[50] B. Vasilescu, A. Serebrenik, M. Goeminne, and T. Mens. On the variation and specialisation of workload—a case study of the gnome ecosystem community. *Empirical Software Engineering*, 2013. accepted.

[51] B. Vasilescu, A. Serebrenik, and M. G. J. van den Brand. By no means: a study on aggregating software metrics. In *Proceedings of the 2nd International Workshop on Emerging Trends in Software Metrics*, WETSoM '11, pages 23–26, New York, NY, USA, 2011. ACM.

[52] B. Vasilescu, A. Serebrenik, and M. G. J. van den Brand. You can't control the unfamiliar: A study on the relations between aggregation techniques for software metrics. In *ICSM*, pages 313–322, 2011.

[53] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[54] D. W. Zimmerman and B. D. Zumbo. Parametric alternatives to the Student t test under violation of normality and homogeneity of variance. *Perceptual and Motor Skills*, 74(3(1)):835–844, 1992.