Modularity analysis of automotive control software

Dajsuren, Y.; van den Brand, M.G.J.; Serebrenik, A.

Published in: ERCIM News

Published: 01/01/2013

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Modularity Analysis of Automotive Control Software

by Yanja Dajsuren, Mark G.J. van den Brand and Alexander Serebrenik

A design language and tool like MATLAB/Simulink is used for the graphical modelling and simulation of automotive control software. As the functionality based on electronics and software systems increases in motor vehicles, it is becoming increasingly important for system/software architects and control engineers in the automotive industry to ensure the quality of the highly complex MATLAB/Simulink control software. For automotive software, modularity is recognized as being a crucial quality attribute; therefore at Eindhoven University of Technology in the Netherlands we have been carrying out industrial case studies on defining and validating the modularity of Simulink models.

This research is part of the Hybrid Innovations for Trucks (HIT) project, which is carried out in a consortium of automotive manufacturer, suppliers, and research institutes, and aims to deliver a significant contribution in realizing up to 7% CO₂ emission reduction and up to 7% fuel saving for long-haul vehicles. Achieving this goal necessitates definition of proper quality techniques to enable the development of more efficient control software. In conventional vehicles, software is used in the multimedia, comfort, and safety systems. However, in (hybrid) electric vehicles, more complex control software for engine, after-treatment, and energy management systems for reducing fuel consumption and harmful emissions are being developed. Automotive control software is being developed using model-based design tools like MATLAB/Simulink and Stateflow. Simulink models can consist of several dozens of subsystems, hundreds of building blocks, and many hierarchical levels, which result in complex and difficult to maintain models. Although there are existing techniques and tools such as Mathworks Automotive Advisory Board (MAAB) guidelines and Model Advisor from Mathworks available to ensure guideline conformance and correctness, techniques for assessing quality of Simulink models are limited.

For automotive software, modularity is recognized as being paramount since changing or reusing non-modular software is very costly. MathWorks provides quality related tools like “Modeling Metric Tool” and “sldiagnostics” 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. Other quality metrics e.g. for measuring instability, abstractness, and complexity of Simulink models have been intro-

References:

Please contact: Javier Ibanez-Guzman
Tel: +33 1 768 57576
E-mail: Javier.ibanez-guzman@renault.com
duced, however the validation of the metrics is not provided. Therefore, we focused on modularity as one of the main quality attributes for Simulink models. According to the ISO/IEC 25010 international quality standard, modularity is defined as the 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. Following the Goal-Question-Metrics (GQM) approach, we defined a modularity metrics suit consisting of nine coupling and cohesion-related metrics for Simulink models and validated it with experts’ evaluation [1]. We identified three independent metrics based on the statistical analysis and identified a relation between modularity metrics and presence of errors. We have observed that high coupling metric values and high number of hierarchical levels (subsystem depth) frequently correspond to presence of errors. We developed a Java tool to measure the defined metrics on Simulink models. The tool uses a Java parser for Simulink MDL files of the ConQAT open-source tool. We are extending our tool to measure not only modularity metrics but also other key quality attributes such as complexity, reusability, testability, and maturity of the Simulink models and extend the visualization of software quality with SQuAVisiT (Software Quality Assessment and Visualization Toolset) toolset [2].

Currently, our metrics tool interfaces with the SQuAVisiT toolset as it is a flexible tool for visual software analytics of multiple programming languages. Figure 1 illustrates an example visualization of the modularity aspect of the industrial application that we studied (the subsystem names here are blurred due to confidentiality reasons). 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. The colours on subsystems are used to visualize values of modularity metrics. The green subsystems show the modular and red ones show subsystems which require attention to improve their modularity. Figure 2 illustrates another example of quality visualization. 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. However, the effectiveness of using different visualizations of Simulink model quality remains a future work.

For more than a decade automotive Architecture Description Languages (ADLs) have been considered as one of the main solutions to represent system and functional architecture and to facilitate communication between different stakeholders. Based on our evaluation of a set of (automotive) ADLs like EAST-ADL, AADL, and SysML against automotive specific modelling requirements, definition of automotive architectural quality is still an open issue [3]. To this end, further work is needed to investigate quality metrics for automotive architectural models.

References:

Please contact:
Yanja Dajsuren
Eindhoven University of Technology,
The Netherlands
Tel: +31 40 247 5052
E-mail: y.dajsuren@tue.nl

Figure 1: Visualization of system modularity with the help of SQuAVisiT tool [2]. This tool gives early feedback about system modularity, making it cheaper and easier to reuse and maintain than traditional techniques. Illustration: TU/e.

Figure 2: Visualization of modularity metric values and number of faults of the subsystems. This can help the domain experts locate easily the most problematic subsystems and their respective metric values. Illustration: TU/e.