1.5 Integrated Design and Configuration of Versatile Software Documents in Automotive Software Engineering

Cem Mengi (mengi@i3.informatik.rwth-aachen.de)
Supervisor: Prof. Dr.-Ing. Manfred Nagl

Software engineering in the automotive domain has gained more and more importance. Today, about 80% of all innovations are software-based. However, due to the traditional hardware-driven development process, automotive software engineering gets highly *complex*. Moreover, the possibility to select optional fittings, e.g., parking assistant, rain sensor, intelligent light system etc., leads to the situation that an enormous number of *software variants* arise.

To overcome the growing complexity, the automotive industry agrees that there is a need to shift from a hardware-driven to a function-driven process, i.e., to abstract from hardware details such as the underlying ECU topology, deployment decisions, real-time characteristics etc. In GRK 643, we provide new methods, concepts, and tools to support the function-driven design process. We distinguish between four levels of models, the *conceptual model*, *behavioral model*, *implementation model*, and *source code model*. The conceptual model describes the first virtual realization of the static system structure and builds a bridge between requirements and architecture specification. The behavioral model is used for simulation of functional behavior. Typically, there are infinite resources available, the model is abstracted from hardware topology, and from any deployment decision. If deployment decisions are made, the behavioral model is enriched with real-time characteristics, the software structure and data types are optimized, and the model is tied to technical interfaces such as bus systems and hardware resources. We call the result of this task as implementation model. Finally, the source code model is generated by the implementation model so that it could be executed on specific hardware platforms.

In order to handle variability in a function-driven development process we provide an integrated approach to handle variation points in the software documents of the different levels. To capture the points of variation, we provide a so called variability model. Thereby, variation points and their variants are organized in a tree-based structure which is extended with constraints in order to express variability such as mandatory, optional, and exclusive variation points. The variability model is integrated into the four levels of models so that it can be used synchronously. Finally, we provide a mechanism to configure the variability model in order to derive specific variants.

[Men10] C. Mengi
Integrated Design and Configuration of Versatile Software Documents in Automotive Software Engineering.
In: Proceedings of the Joint Workshop of the German Research Training Groups in Computer Science,
Verlagshaus Mainz GmbH Aachen, ISBN 3-86130-146-6, January 2010.
www.se-rwth.de/publications