Demonstrating Development of Software Architecture of Multi-core Real-time Vehicular Systems

Alessio Bucaioni*†, Federico Ciccozzi*, Antonio Cicchetti*, Saad Mubeen†*, Mikael Sjödin*,
Mattias Gålnander†, John Lundbäck† and Kurt-Lennart Lundbäck†
* *Mälardalen Real-Time Research Centre (MRTC), Mälardalen University, Västerås, Sweden*
† *Arcticus Systems AB, Järfälla, Sweden*
*{*name.surname*}*@mdh.se*
†{*name.surname*}*@arcticus-systems.com*

***Abstract*—We present a demonstrator for the model- and component-based development of the software architecture of multi-core real-time vehicular systems using the industrial modelling language Rubus Component Model and its integrated development environment Rubus-ICE. We demonstrate various stages of the development process such as modelling of the software architecture, automatic generation of code, simulation and testing.**

I. Background – The Rubus Concept

Due to the need of higher computational power [1] and safety [2], always more vehicular embedded systems are realised by means of parallel platforms notably multi-core ones. In this scenario, it is paramount that development strategies for vehicular embedded software and the supporting tools are able to efficiently deal with the multi-core specific challenges such as, separation of software from hardware modelling, support for software to hardware allocation, etc. Rubus is a collection of methods, theories and tools for the model- and component-based development of predictable, timing analysable and synthesisable control functions in resource-constrained embedded systems [3] [4]. Rubus is developed by Arcticus Systems AB [1] in close collaboration with Mälardalen University. Through the years, it has been adopted by several Original Equipment Manufacturer (OEM), Tier-1 and Tier-2 companies (such as Volvo Construction Equipment [2], BAE Systems Hägglunds [3], Hoerbiger [4], etc.) for the development of vehicular embedded software. The Rubus concept is based around the Rubus Component Model (RCM), a domain-specific modelling language used for representing the software functions, the hardware platform, the software to hardware allocation and the real-time properties of the vehicular embedded software under development. The Rubus concept features a complete development environment, Rubus-ICE , which includes the following:

- Designer: A graphical modelling tool based on RCM. It creates a set of XML-files containing the software architecture ad the deployment information related to selected Run-Time Environment (RTE) and target.
- Analyzer: A graphical off-line and on-line analysis tool. The off-line analysis includes tasks and network messages response time analysis, shared stack analysis and end-to-end distributed response-time and delay analysis [5]. The on-line analysis reads execution traces from the target environment via a communication channel.
- Inspector: A graphical testing tool for software- and hardware-in-the-loop testing.
- Simulator: A graphical simulation environment for controlling the execution of the embedded software from high-level simulation tools such as LabView, Simluink, etc.
- Build tools: compiler, linker, and plug-ins launcher.
- Synthesizer: A code-generation tool which generates the execution framework for a specific RTE-platform.

II. Demonstration of Development Process

We demonstrate the applicability of RCM and the usage of Rubus-ICE by modelling a distributed real-time vehicular application on multi-core. The vehicular application consists of two nodes running the Rubus operating system and connected via a Controller Area Network (CAN) and is inspired by industrial applications. We demonstrate the following steps during the development.

1) Modeling: Designing of the software architecture, hardware platform and software to hardware allocation of the modelled vehicular application.

2) Analysis: Performing different types of analysis available in Rubus-ICE such as the end-to-end response-time and delay analysis and stack-memory analysis.

3) Synthesis: Automatic code-generation for the run-time infrastructure (execution framework).

4) Simulation and Testing: Controlled execution of the modelled vehicular application in a simulated environment from Simulink. Testing of the modelled vehicular application at various hierarchical levels.

[1] http://www.arcticus-systems.com
[2] https://www.volvoce.com
[3] http://www.baesystems.com/en/home
[4] https://www.hoerbiger.com

REFERENCES

[1] R. N. Charette, "This car runs on code," *IEEE Spectrum*, vol. 46, no. 3, p. 3, 2009.

[2] ISO 26262-1:2011: Road Vehicles in Functional Safety. http://www.iso.org/.

[3] "Rubus ICE-Integrated Development Environment," http://www.arcticus-systems.com.

[4] "Rubus models, methods and tools," http://www.arcticus-systems.com.

[5] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Support for end-to-end response-time and delay analysis in the industrial tool suite: Issues, experiences and a case study," in *Computer Science and Information Systems, vol. 10, no. 1, pp 453-482, January 2013*.