Demonstrator for modeling and development of component-based distributed real-time systems with Rubus-ICE

Alessio Bucaioni*, Saad Mubeen†*, John Lundbäck†, Kurt-Lennart Lundbäck†, Jukka Mäki-Turja†* and Mikael Sjödin*
* *Mälardalen Real-Time Research Centre (MRTC), Mälardalen University, Västerås, Sweden*
† *Arcticus Systems AB, Järfälla, Sweden*
*{*alessio.bucaioni, saad.mubeen, jukka.maki-turja, mikael.sjodin*}@mdh.se*
†{*saad.mubeen, john.lundback, kurt.lundback*}@arcticus-systems.com*

***Abstract*—We present a demonstrator for modeling and development of component-based vehicular distributed real-time systems using the industrial model Rubus Component Model (RCM) and its development environment Rubus-ICE (Integrated Component development Environment). It demonstrates various stages during the development process of these systems such as modeling of software architecture, performing timing analysis, automatic synthesis of code from the software architecture, simulation, testing, and deployment.**

I. Background – The Rubus Concept

Development strategies for real-time embedded systems in the automotive and other vehicular applications domain are to an extent based on model- and component-based development approach. This approach uses models to describe functions, structures and other design artifacts; and supports the development of large software systems by integration of software components. It raises the level of abstraction for software development and aims to reuse software components and their architectures. Rubus [1], [2] is a collection of methods, theories and tools for model- and component-based development of predictable, timing analyzable and synthesizable control functions in resource-constrained embedded systems. Rubus is developed by Arcticus Systems in close collaboration with Mälardalen University and is used by several international companies. The Rubus concept is based around RCM and Rubus-ICE which includes the following.

- Designer: A graphical tool for modeling a system based on RCM. It creates a set of XML-files containing the design including 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 consists of response-time analysis of tasks and network messages, shared stack analysis, and end-to-end distributed response-time and delay analysis [3]. Whereas, the on-line analysis reads execution trace from the target via a communication channel. The Rubus Analyzer gives a possibility to feed back information from the target.
- Inspector: A graphical component test tool for software as well as hardware in the loop tests.
- Simulator: It builds a simulated environment around the application to allow the control of its execution from a high-level tool such as LabView or Matlab/Simulink.
- Build tools: compiler, linker, and plug-ins launcher.
- Synthesizer: A tool to generate the execution framework for a specific RTE-platform.

An example of software architecture modeled in RCM is shown in Figure 1. The organization and screen shots of Rubus-ICE are shown on the next page.

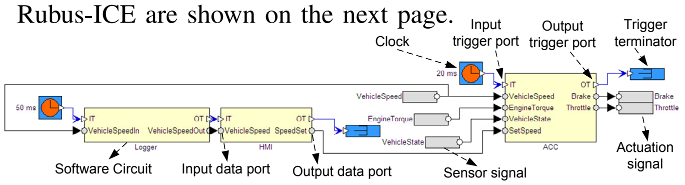


Figure 1. Architecture of a system modeled in RCM

II. Demonstration of Development Process

We demonstrate the methodology and usage of Rubus tools by modeling and developing a distributed real-time application which is the simplified Intelligent Parking Assist System. It consists of two nodes that run the Rubus operating system and are connected via Controller Area Network. We demonstrate the following steps during the development.

1) Modeling: Developing component-based software architecture of the application with Rubus modeling language.

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: Automatically generating the code for the run-time infrastructure (execution framework).

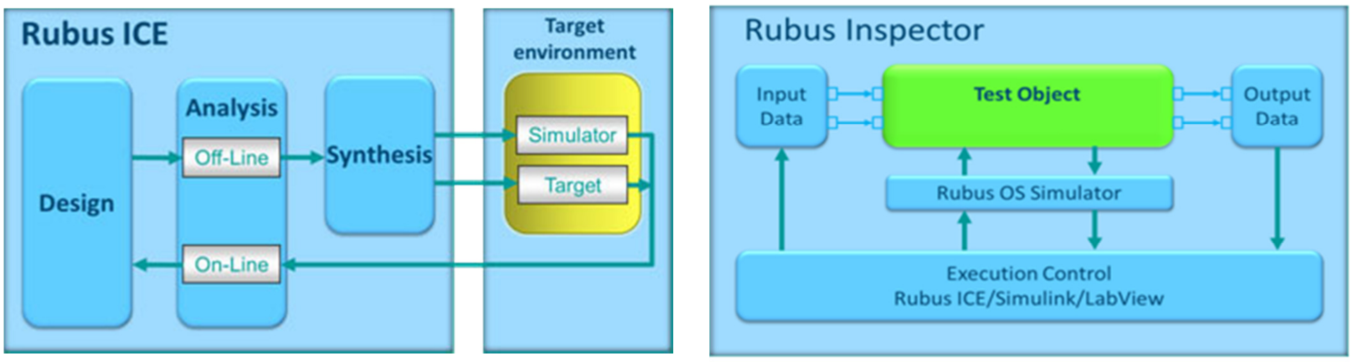
4) Simulation and Testing: Executing the modeled application in a simulated environment from Simulink and testing at various hierarchical levels.

5) Deployment: Downloading the synthesized software on hardware platform and demonstrating the functionality.

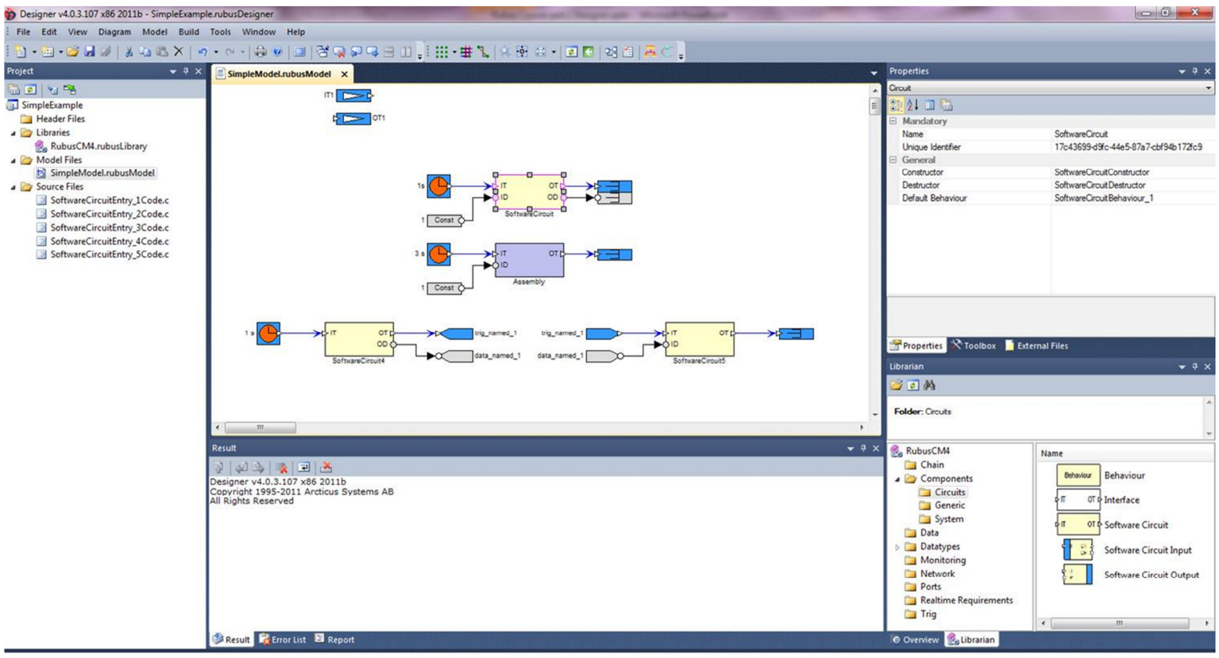
References

[1] "Rubus models, methods and tools," http://www.arcticus-systems.com.
[2] K. Hänninen et.al., "The Rubus Component Model for Resource Constrained Real-Time Systems," in *3rd IEEE International Symposium on Industrial Embedded Systems*, June 2008.
[3] 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," *Computer Science and Information Systems*, vol. 10, no. 1, 2013.

Screen Shots of the Tools



Organization of Rubus-ICE

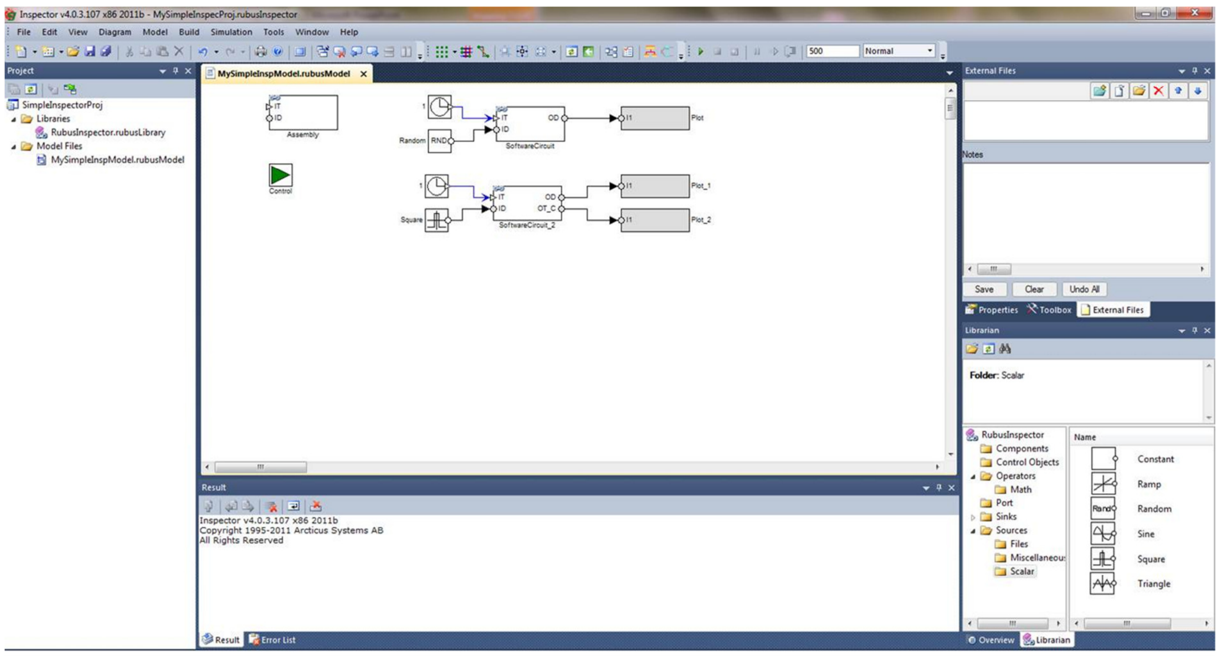


Circuit

Mandatory	
Name	SoftwareCircuit
Unique Identifier	17c43699-d9fc-44e5-87a7-cbf94b172fc9
General	
Constructor	SoftwareCircuitConstructor
Destructor	SoftwareCircuitDestructor
Default Behaviour	SoftwareCircuitBehaviour_1

```
Designer v4.0.3.107 x86 2011b
Copyright 1995-2011 Arcticus Systems AB
All Rights Reserved
```

Rubus Designer



```
Inspector v4.0.3.107 x86 2011b
Copyright 1995-2011 Arcticus Systems AB
All Rights Reserved
```

Rubus Inspector