Academic-industrial Collaboration in the Vehicle Software Domain: Experiences and End-user Perspective

Saad Mubeen, Jukka Mäki-Turja, John Lundbäck, Mattias Gålnander, Kurt-Lennart Lundbäck, Mikael Sjödin, Bud Lawson

Academic-industrial Collaboration in the Vehicle Software Domain: Experiences and End-user Perspective

Saad Mubeen*, Jukka Mäki-Turja*, John Lundbäck†, Mattias Gålnander†, Kurt-Lennart Lundbäck†, Mikael Sjödin*, Harold "Bud" Lawson‡
* Mälardalen Real-Time Research Centre (MRTC), Mälardalen University, Västerås, Sweden
† Arcticus Systems AB, Järfälla, Sweden
‡ Lawson Konsult AB, Lidingö, Sweden
*{saad.mubeen, jukka.maki-turja, mikael.sjodin}@mdh.se
†{john.lundback, mattias.galnander, kurt.lundback}@arcticus-systems.com
‡bud@lawson.se

***Abstract*—In this paper we present a success story of academic-industrial collaboration in the vehicular domain. The collaboration has resulted in the development and evolution of a model- and component-based software development tool chain that is used to develop safety-critical, robust, and certified, control software for vehicles. The tool chain has been successfully used in the vehicle industry for about 20 years. The sustained development of the tool chain is based one a unique collaboration, described in this paper, where the collaborators form a clear value chain from academia, through tool-developer, to the end-users of the technology. We describe experiences of each collaborator with a focus on the end-user's perspective. Moreover, we highlight some ongoing and future works within this collaboration.**

I. Introduction

A large share of innovation and customer value in modern vehicles comes from advanced computer-controlled functionality. With the increasing volume of such functionality, the vehicle software has tremendously increased in size and complexity in the past few years [1], [2]. The distributed nature of this software over several Electronic Control Units (ECUs) further adds to its complexity. Moreover, the safety-critical nature of several vehicle functions puts real-time requirements on them. The developers of such functions are required to ensure that the functions are predictable, i.e., they behave in a timely manner when executed. The software complexity can be managed by using the tools that employ the principles of component-based software engineering [3], [4] and model-driven engineering [5]. On the other hand, predictability of the functions is guaranteed by the tools that implement real-time schedulability analysis [6], [7], [8]. Such analysis can validate the timing requirements, without performing exhaustive testing, before the functions are deployed in the target platforms.

In this paper, we consider the case of one such tool chain: Rubus [9]. The Rubus tool chain supports model- and component-based development of vehicle software and its predictable execution on a real-time operating system (RTOS) that is certified to ISO 26262:2011 safety standard according to ASIL D. In this paper, we describe the experiences of various collaborators in the academic-industrial collaboration that has resulted in the development of the Rubus tool chain and its evolution based on the state-of-the-art research results, industrial needs and feedback from the end-users. One unique characteristic of the selected collaboration is that it offers a clear value chain from academia (mainly Mälardalen University); through tool developer (Arcticus Systems[1]); and finally, to the end users of the technology (e.g. Volvo Construction Equipment[2] and BAE Systems Hägglunds[3]) as shown in Fig. 1. This figure shows the flow of information within the selected academic-industrial collaboration.

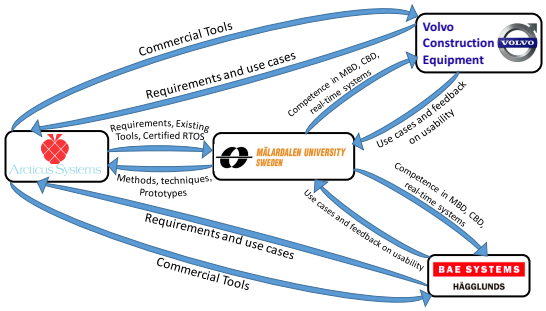

Fig. 1. Example of flows of information within the collaboration.

II. The Rubus Tool Chain and Perspective of the Tool Provider

There is a large number of tool providers in the vehicle domain such as Arcticus Systems, Vector, Symta Vision, Systemite, Mentor Graphics, IBM, Arccore, ETAS, Continental, dSpace, Berner & Mattner, Fraunhofer, just to name a few. In this paper, we consider the case of the Rubus tool chain that is developed by Arcticus Systems in close collaboration with several academic and industrial partners. The main reasons for selecting Rubus are as follows. The software developed using Rubus has a small run-time foot print (timing and memory overhead) as compared to several other component-based development technologies including AUTOSAR [10]. While most of the tool chains and models need complementary tools from other vendors to support the software development at all abstraction levels of EAST-ADL [11], the Rubus tool chain is applicable to all the abstraction levels. A detailed comparison of the Rubus tool chain with other related models and tool chains is discussed in [12]. Arcticus Systems and their customers are relatively open in discussing the theories

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

and details about their tools, use cases and experiences with the research community. Finally, the Rubus tool chain has been successfully used in the vehicle industry for about 20 years.

The main products in the Rubus tool chain include Rubus-ICE (Integrated component model Development Environment) and Rubus RTOS as portrayed in Fig. 2. A real-time application developed using Rubus-ICE can be executed on a variety of platforms, that is, various hardware and real-time operating systems. Rubus-ICE is utilized by several international companies including Volvo Construction Equipment, Volvo Group, BAE Systems Hägglunds, Elektroengine, BorgWarner, Hoerbiger and Knorr-Bremse in developing, simulating and implementing time-critical and non-time critical applications. Rubus provides for the implementation of the model-driven development concept by providing the following properties.

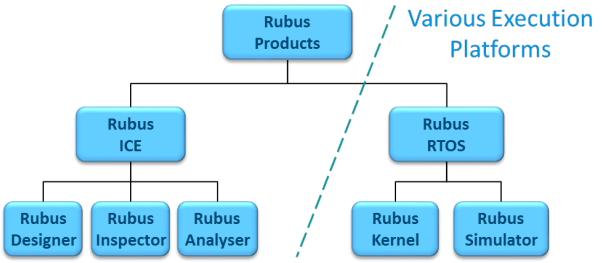

Fig. 2. The Rubus tool chain.

- Raises the level of abstraction thus addressing the increasing complexity problem for embedded software.
- *Formal and early reasoning.* The system architecture can be described early in the system life cycle as high-level models and alternative designs can be rapidly developed and analyzed to try out different solutions. Note that the design can be analyzed without writing any source code. Furthermore, system model documentation facilitates maintenance and further development activities.
- *Code synthesis.* There is a separation between the run-time model and the program code. The user works on a platform independent model, then selects the specific target platform, and Rubus-ICE generates the framework code. Productivity is increased since the auto-generation automates code generation that is often error prone.
- *Traceability.* The system architecture documentation is kept up to date with the implementation resulting in traceability from design to implementation and vice versa.

The Rubus Designer is used to interactively describe and analyse the application developed using the Rubus Component Model (RCM)[13]. The structure of the application is developed by means of Software Circuits (SWCs), lowest-level hierarchical components, and their interconnections. The user defines the input and output ports of SWCs and connects them in a manner similar to hardware diagrams.

The Rubus Inspector supports platform-independent formal and semi-formal "Model-in-the-Loop testing environment. It provides for unit testing (SWCs and assemblies, i.e., containers of SWCs) as well as for sub-system testing (node and network) or complete system testing (distributed system with multiple nodes and networks). Test inputs are provided by the user however, it is also possible to generate tests that utilize LabView/Simulink and Matlab environments.

The Rubus Analyser provides for the user-friendly presentation of off-line and on-line information about the execution behavior of the system. It enables connection with the development host system to download information such as trace data and run-time information in real-time. Post-run-time analysis of host data for formal analysis can be compared with the original model data to verify the real-time properties initially set by the designer. Another powerful characteristic of the Rubus Analyser is its support for pre-run-time timing analysis including response-time analysis of tasks with offsets [14], shared-stack analysis [15], response-time analysis of Controller Area Network (CAN) and its higher-level protocols [16], distributed end-to-end delay analysis [17], [7], as well as early timing analysis of black box nodes (whose internal software architectures are not available) [18].

Rubus RTOS. The Rubus tool chain is complemented by Rubus RTOS that is certified in ISO 26262:2011 safety standard. Rubus Kernel provides support for RCM in achieving an optimal real-time software system. Its main features include support for both time- and event-triggered execution of threads, inter-thread communication, static allocation of resources, scalability and portability. The combination of a dynamic and static scheduling supported by the Rubus Kernel enables the design of optimal real-time software systems. The kernel can be ported to various targets and development environments including Freescale MPC-processors, Texas DSP, Infineon's xc167-processors and various C-compiler environments such as Green Hills, WindRiver, Tasking, Microsoft VS and GCC. Rubus Simulator provides for testing and verifying the composite of the Rubus Kernel and application software.

III. Academic Perspective

Since the conception of the first Rubus products, cooperation with Mälardalen University (and other academic institutions) has been of great importance to drive the development of the Rubus tool chain. Vice versa, experience and feedback from industrial use of Rubus have inspired many research projects over the years. A graphic history of different component models that have been developed in academia and by Arcticus Systems is shown in Fig. 3.

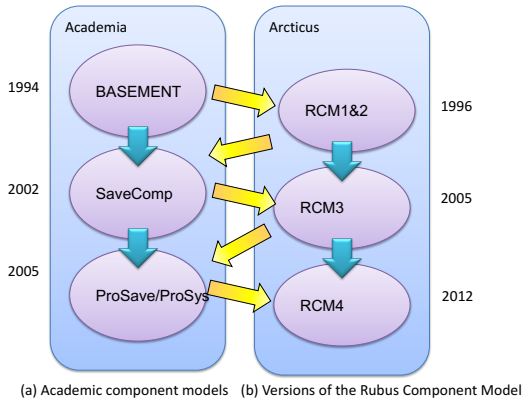

Fig. 3. Contributions to the Rubus Component Model and tool chain.

Rubus was initially developed from the joint academic and industrial project VIA, project BASEMENT and its conceptual component model [19]. Later, an SSF-funded project, SAVE, coordinated by Mälardalen University with partners from KTH, Linköping University, and Uppsala University based the SaveComp component model on the BASEMENT-concept and the experiences from early Rubus models. In a technology-transfer project MULTEX, funded by KKS, Mälardalen University worked with Arcticus Systems to define

the next version of RCM using concepts from SaveComp. This resulted in the first version of Rubus where components could be used in time-, event- and interrupt-triggered threads. In 2006, a KKS-funded project at Mälardalen University, called PROGRESS, provided the ProCom component model [20] that inspired further development of RCM. A new technology-transfer project, EEMDEF, funded by KKS resulted in the extension of RCM and Rubus tool chain to support the modeling and development of distributed embedded systems [12].

Another KKS-funded project FEMMVA resulted in the extension of timing analysis framework in Rubus tool chain to support the specification, analysis and validation of end-to-end path delay constraints namely age and reaction [17], [7], [10]. Further, the results of the project provided foundations for the translation of timing constraints and design-level models from EAST-ADL to RCM. Since 2012, and the finalization for Rubus Component Model Version 4.0, Arcticus has further intensified the academic cooperation with Mälardalen University and with participation in several larger European projects. Many projects are a result of the continued cooperation with Mälardalen University. Finally, a Worst Case Execution Time (WCET) analysis project done in cooperation with Mälardalen University provided an important step toward adding WCET analysis of the C-source code as a complement to measurement on target processors.

IV. Perspective of the End User

We consider two end-users of the Rubus tool chain as part of the selected collaboration.

A. BAE Systems Hägglunds

BAE Systems is a renowned global manufacturer of advanced defense, combat and security systems that can be used in the air, land and sea. BAE Systems Hägglunds excels at manufacturing state-of-the-art combat vehicles. According to BAE Systems Hägglunds, the main reasons for selecting RCM and Rubus tool chain for the development of embedded software in their products are as follows.

- **Requirement for the time- and event-triggered execution:** They require time-triggered architectural approach that is needed to execute the software threads that compose majority of the repetitive functions in their products. The support for event-triggered execution of threads is needed to service various internal and external events in the system. Both the time- and event-triggered modeling and execution of software circuits are fully supported by RCM, Rubus-ICE and Rubus RTOS.
- **Requirement to support background threads and services:** They require the support to implement background threads that are appropriate for services of diverse character such as diagnostics as well as vital functions for the implementation of communication protocols such as TCP/IP. This requirement is fully supported by the Rubus tool chain.
- **Requirement of pre-runtime timing analysis:** They require pre-runtime guarantees on the predictability of both time- and event- triggered threads. Using RCM and the Rubus tool chain, they provide pre-runtime guarantees for time-triggered software circuits by using the correct-by-construction offline schedule that is generated by the Rubus scheduler. Whereas in the case of event-triggered software circuits, response-time analysis and end-to-end delay analysis in Rubus-ICE provides pre-run-time guarantees on the timing behavior of the software circuits.
- **Requirement of coupling the software components with Simulink:** The behavior of control functions are developed using Simulink. There is a requirement to couple each software component in the software architecture with Simulink. The coupling to Simulink is achieved at a suitable level via a modified code generator such that the generated code becomes an entry point for a software circuit. The software circuit is then allocated to a time-triggered chain with an appropriate periodicity.

After their adoption of the Rubus tool chain, they have provided the industrial needs, requirements and use cases during the evolution of the Rubus tool chain. Moreover, they have provided evaluation of the methods, techniques, results as well as corresponding extensions in the Rubus tool chain in a realistic industrial environment.

B. Volvo Construction Equipment (VCE)

VCE is the oldest and one of the world-leading manufacturers of heavy construction-equipment vehicles, e.g., wheel loaders, articulated haulers and excavators. The company has been successfully applying model-based software development, using the Rubus tool chain, to provide computer-controlled functionality in the vehicles since 1997. VCE has used Rubus in almost all of their products, most notably, for controlling the gear shift and various other ECUs. According to VCE, Rubus has proven to work efficiently from small applications with only a few objects up to very large applications with thousands of objects. Some of their experiences are listed below.

- **Development and testing times:** Small development time is of essence for VCE. The component concept in Rubus offers advantages such as divide-and-conquer strategies, i.e., breaking down of an application into smaller components that can be distributed to a large number of developers. This enables development of vehicle functionality in parallel, hence resulting in shortening the development time. The support for model-in-the-loop testing and creation of unit tests for each component in the Inspector, one of the tools in Rubus-ICE, has resulted in reduction of the amount of bugs that end up during the integration of a system. As a result, time to test has been shortened after using the Rubus tool chain.
- **Support for interoperability:** Interoperability of information and models during the development of vehicle software is very important for VCE. Such a support is provided by the Rubus tool chain since it allows integration with other tools such as Simulink/Matlab. Using Rubus, there are various means of either using the application programming interfaces to read/write data or directly access the model information using scripts such as Python.
- **Information management:** Rubus tool suit uses XML-files as an exchange format. Such a format enables the use of existing change management systems such as SVN, ClearCase and GIT. Merging is also facilitated since the information is human readable but still possible for a machine to process. It also means that creating model information from other tools or means is relatively easy. Hence, information management becomes easier.
- **Pre-runtime timing analysis:** A lot of functionality in the vehicles, produced by VCE, have real-time require-

ments. VCE is required to verify such requirements before the functionality is deployed in the vehicles. The pre-runtime timing analysis support in Rubus-ICE, including response-time analysis and distributed end-to-end delay analysis, has proven to yield good results in respect to early verification, feasibility validation and timing checks.
- **Certification:** VCE needs to deliver certified vehicle functionality according to safety standards. Since the Rubus RTOS is certified according to ISO-26262:2011 ASIL-D, it serves the purpose for VCE. The Rubus tool suite is currently undergoing the certification process.

VCE has contributed to the evolution of the Rubus tool chain by not only supplying industrial needs, requirements and use cases but also providing the efficacy of the new methods, techniques, results and corresponding extensions.

V. Summary

In this section we summarize our experiences by a set of quotes from key stake-holders in our collaboration.

- "In order to be competitive as a tool provider in the embedded systems domain, we strive to be active, agile and fast in identifying the needs and challenges of our customers and provide them solutions based on the state-of-the-art techniques and research results" (Arcticus Systems).
- "To remain academically excellent, and scientifically relevant, we find the long-term cooperation within the team highly invigorating. Over the years the collaboration have had a profound impact on our whole research environment at MDH" (Prof. Mikael Sjödin, Mälardalen University).
- "Based upon my long-term cooperation with Arcticus Systems, I am pleased with their work towards implementing a very useful real-time kernel and supporting tool suite. It has been personally satisfying since Arcticus Systems has made usage of several of the concepts that I previously developed" (Harold "Bud" Lawson).
- "From the end-user's perspective some remarkable features of Rubus include user-friendly tools, simplicity to understand and use by the engineers, light run-time foot print and automation" (VCE).
- "The good thing about Rubus is that it supports both time- and event-triggered execution. With the time-triggered execution, majority of our functions are supported well due to their repetitive nature. The communication and diagnostic services in our systems are well supported by utilizing the event-triggered execution in Rubus. Interoperability between Rubus and Simulink eases our effort for the development of control logic" (BAE Systems).

"Automated interoperability to support information exchange among tools and models during the development of cyber-physical systems is going to be the next big thing in the vehicle industry (industrial members of the collaboration)."

VI. Ongoing Work

In modern vehicles, contemporary functions and advanced features require new levels of computational power and more complex coordination among subsystems. Multi-core platforms offer an efficient support for running such computation-intensive vehicle functions by executing various activities in parallel. However, multi-core platforms are more prone to unpredictable behavior as compared to the single-core platforms. How to provide predictable execution of vehicle software in parallel on a contemporary multi-core ECU with shared caches and memory banks? It is still an unsolved problem. This problem is expected to increase in the future with the introduction of more advanced architectures with cluster sets of cores having more advanced memory interconnects. A seamless tool chain for model- and component-based development of vehicle software its predictable execution on multi-core platforms is missing in the state of the practice. Currently, the selected collaboration is working to achieve these goals in the ongoing projects[4] such as EMC^2, DPAC, ASSUME and PreView. Another ongoing work includes certification of parts of the tool chain according to the ISO 26262 safety standard.

References

[1] M. Broy, I. Kruger, A. Pretschner, and C. Salzmann, "Engineering automotive software," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 356 –373, feb. 2007.

[2] R. N. Charette, "This Car Runs on Code," *Spectrum, IEEE*, vol. 46, no. 2, 2009, http://spectrum.ieee.org/green-tech/advanced-cars/this-car-runs-on-code.

[3] I. Crnkovic and M. Larsson, *Building Reliable Component-Based Software Systems*. Norwood, MA, USA: Artech House, Inc., 2002.

[4] T. A. Henzinger and J. Sifakis, "The Embedded Systems Design Challenge," in *14th International Symposium on Formal Methods (FM), Lecture Notes in Computer Science*. Springer, 2006, pp. 1–15.

[5] D. C. Schmidt, "Guest editor's introduction: Model-driven engineering," *Computer*, vol. 39, no. 2, pp. 25–31, Feb. 2006.

[6] N. Audsley, A. Burns, R. Davis, K. Tindell, and A. Wellings, "Fixed priority pre-emptive scheduling: an historic perspective," *Real-Time Systems*, vol. 8, no. 2/3, pp. 173–198, 1995.

[7] 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.

[8] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis - the symta/s approach," *Computers and Digital Techniques*, vol. 152, no. 2, pp. 148–166, March 2005.

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

[10] "AUTOSAR Techincal Overview, Release 4.1, Rev. 2, Ver. 1.1.0., The AUTOSAR Consortium, Oct., 2013," http://autosar.org.

[11] "EAST-ADL Domain Model Specification, V2.1.12,," http://www.east-adl.info/Specification/V2.1.12/EAST-ADL-Specification_V2.1.12.pdf.

[12] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Communications-Oriented Development of Component- Based Vehicular Distributed Real-Time Embedded Systems," *Journal of Systems Architecture*, vol. 60, no. 2, pp. 207–220, 2014.

[13] 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.

[14] J. Mäki-Turja and M. Nolin, "Tighter response-times for tasks with offsets," in *Real-time and Embedded Computing Systems and Applications Conference (RTCSA)*, August 2004.

[15] M. Bohlin, K. Hänninen, J. Mäki-Turja, J. Carlson, and M. Sjödin, "Bounding shared-stack usage in systems with offsets and precedences," in *20th Euromicro Conference on Real-Time Systems*, July 2008.

[16] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Integrating Mixed Transmission and Practical Limitations with the Worst-Case Response-Time Analysis for Controller Area Network," *Journal of Systems and Software*, 2014.

[17] N. Feiertag, K. Richter, J. Nordlander, and J. Jonsson, "A Compositional Framework for End-to-End Path Delay Calculation of Automotive Systems under Different Path Semantics," in *Workshop on Compositional Theory and Technology for Real-Time Embedded Systems*, dec. 2008.

[18] S. Mubeen, M. Sjödin, T. Nolte, J. Lundbäck, M. Gålnander, and K.-L. Lundbäck, "End-to-end Timing Analysis of Black-box Models in Legacy Vehicular Distributed Embedded Systems," in *21st International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, Aug. 2015.

[19] H. Hansson, H. Lawson, O. Bridal, C. Eriksson, S. Larsson, H. Lon, and M. Stromberg, "Basement: an architecture and methodology for distributed automotive real-time systems," *IEEE Transactions on Computers*, vol. 46, no. 9, pp. 1016–1027, Sep 1997.

[20] S. Sentilles, A. Vulgarakis, T. Bures, J. Carlson, and I. Crnkovic, "A Component Model for Control-Intensive Distributed Embedded Systems," in *CBSE 2008*, pp. 310–317.

[4] http://www.arcticus-systems.com/research/