Provisioning of Deterministic and Non-Deterministic Services for Vehicles: The Rubus Approach

Harold “bud” Lawson, Saad Mubeen, Alessio Bucaioni, Jukka Mäki-Turja, John Lundbäck, Mattias Gålnander, Kurt-Lennart Lundbäck, Mikael Sjödin

Provisioning of Deterministic and Non-Deterministic Services for Vehicles: The Rubus Approach

Harold "Bud" Lawson[‡], Saad Mubeen[*], Alessio Bucaioni[*†], Jukka Mäki-Turja[*], John Lundbäck[†], Mattias Gålnander[†], Kurt-Lennart Lundbäck[†], Mikael Sjödin[*],
[*] 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, alessio.bucaioni, mikael.sjodin}@mdh.se
[†]{john.lundback, mattias.galnander, kurt.lundback}@arcticus-systems.com
[‡]bud@lawson.se

***Abstract*—Providing computer-based services for vehicle functions has evolved to the point where a large majority of functions are realized by software. However, the need to provide safety and security in critical functions such as braking, steering, motor control, etc. requires an approach that can guarantee the continuous reliable operation of the functions. At the same time, there are a variety of functions that are less critical from the vehicle operation perspective that can be provided where safety and security are less critical. From a vehicle manufacturers point of view, providing both types of services in an economic and reliable manner is a real challenge. To meet this challenge, we consider the Rubus Tool Suit for the software development and a well-proven (in industrial use for over twenty years) and certified (according to ISO 26262) operating system Kernel for its execution. In addition, a user-friendly approach to model- and component-based development concept called the software circuits has provided an approach to meet the demands of both safety-critical deterministic and as well as non-safety critical non-deterministic services. In this paper, a brief history of the evolution of Rubus approach as well as an overview of the driving concepts used in providing the Rubus products are described.**

I. History and Evolution of Rubus

Arcticus Systems AB was established as a Swedish corporation in 1985. The Rubus[1] Kernel was introduced for industrial use in 1996. The concepts utilized in Rubus evolved from previous experiences with Arcticus and earlier product OTool as well as their participation in the Swedish Development Agency (NUTEK) VIA (Vehicle Internal Architecture) project during 1992-95. The partners in the project included, in addition to Arcticus, Saab Automotive, Volvo, Mecel AB, SICS, Uppsala University, Chalmers University and Lawson Konsult AB.

Earlier experiences with time-driven deterministic execution in Automatic Train Control (ATC) were used as a primary input to the VIA project. Harold "Bud" Lawson was the architect of the on-board system provided by ITT Standard Radio to the Swedish Railways (SJ). This product was developed during the latter part of the 1970s and has been installed in the majority of locomotives in Sweden starting in 1981. The program code was surprisingly minimal due to the utilization of the hardware like paradigm used in its development (became known as Software Circuits (SWC)). The time determinism and the small amount of program code led to significant advantages in exhaustive testing and verification of the production units. The results of this product development have been reported in [1], [2], [3].

The VIA project identified the need for a mix of time-driven (became known as Red) and event-driven (became known as Blue) real-time tasks. As a central part of the VIA project, the Rubus Kernel was developed to support a mix of both forms of real-time application tasks. Red application tasks are scheduled periodically and have guaranteed execution time slots (deterministic). Blue application tasks are executed within residual time in the time slots (non-deterministic). The VIA project based on the use of the Rubus Kernel and a proposed time deterministic network communication defined a distributed real-time infrastructure for vehicles called BASEMENT. The results of the project where reported in two articles [4], [5].

Arcticus established an agreement with Mälardalen University in 1997 and further research and development led to the development of an off-line scheduler by Christer Norström (formerly Eriksson) now CEO at SICS (Swedish Institute for Computer Science) and Kristian Sandström [6]. This effort formed the basis for the eventual Model Driven Development (MDD) product.

A major breakthrough for Arcticus in 1996 was the selection of the Rubus Kernel, as the real-time operating system, for the implementation of a Limited Slip Coupling Device (for four wheel drive vehicles) by Haldex in Landskrona (now owned by BorgWarner Torq Transfer Systems). After several prototype iterations the first LSCD was delivered to Volkswagen. Since then it has evolved into a significant product that is utilized in most of all four-wheel drive vehicles in the world.

[1]Rubus (the name of a red arctic berry and a registered trademark of Arcticus AB)

In 1997, both Volvo Construction Equipment (VCE) in Eskilstuna and BAE Hägglunds in Örnsköldsvik decided to deploy the Rubus Kernel in their products. These customers' active cooperation with Arcticus led to the development and deployment of the MDD products (Rubus tool suite).

VCE has deployed the Rubus Kernel in several of its products and the Kernel has been certified, in 2015, according to the international standard ISO 26262:2011 with respect to ASIL D (Road Vehicles Functional Safety). As a part of this certification, Arcticus has developed its Quality Management System (QMS) based upon ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207 on Systems, respective Software Life Cycle Processes.

As a result of the long-term cooperation between Arcticus and Lawson Konsult AB, an article describing their contributions to Highly Reliable Real-Time Systems was presented at the 3rd Nordic Conference on Computing [7].

The cooperation between Arcticus and Mälardalen University has continued and is on going where both faculty members and doctoral students have made significant contributions to the Rubus tool suite. In addition to the on-line scheduler, improvements to component models [8], [9], shared stack analysis [10], response-time analysis of singe-node (uniprocessor) systems [11], response-time analysis of Controller Area Network and its higher-level protocols [12], end-to-end response time and delay analysis for distributed systems [13] and worst-case execution analysis. In order to better support interoperability between Rubus and other related models and languages such as EAST-ADL [14], metal-model of the Rubus Component Model (RCM) has been formally defined [15]. In addition, model-based techniques have been employed for realizing a model-based design exploration methodology for easing the transition between the EAST-ADL and the RCM [16], [17].

II. Driving Concepts

Based upon the evolution of Arcticus products, the following central concepts have evolved.

A. Software Circuits

A hardware analogy where component data and control flow behave as a chain of circuits. This promotes the analysis and verification of application function timing constraints and resource utilization that is accomplished by clearly separating data and control flow mechanisms in a network of software circuits. The model of a software circuit is shown in Fig. 1. It is the basic building block and the lowest-level hierarchical element in RCM.

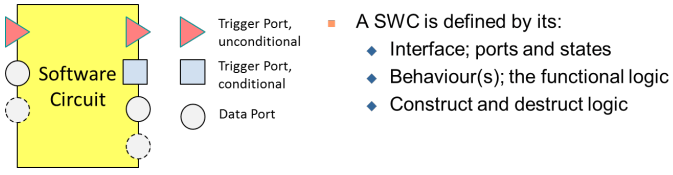

Fig. 1. Model of a Software Circuit.

The Construct and Destruct logic are terms from the object-oriented languages that describe activation, respectively, deactivation of the circuit.

B. Model Driven Development

MDD has had an increasingly important role in designing and implementing real-time embedded systems. Due to the complexity of real-time systems, the development must rely more and more upon automation and the interoperability amongst models such as Simulink. The Rubus Tool Suite provides an integrated tool-chain that includes system modelling, design, analysis and synthesis providing the features portrayed in Fig. 2.

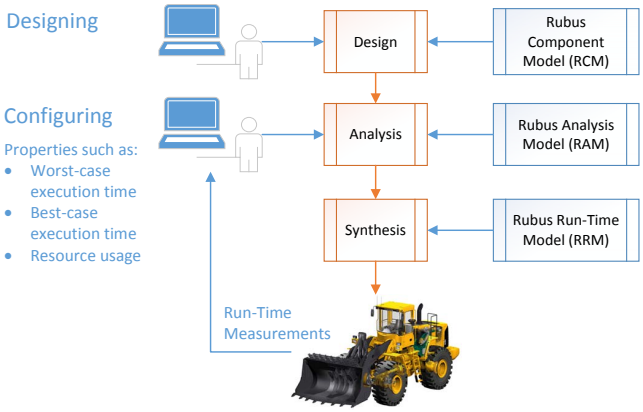

Fig. 2. Rubus Conceptual Models.

The three models provide various viewpoints reflecting all of the necessary information concerning the development, analysis, synthesis and execution of real-time applications.

- **Rubus Component Model (RCM) - Viewpoint of the developer/development team model:** The developer designs the system, in a platform independent manner that focuses upon the application. Timing and resource constraints are expressed in the model.
- **Rubus Analysis Model (RAM) - Viewpoint of the analysis model:** The resulting RCM design is formal and lends itself to static analysis that is mapped to the actual run-time platform. The analysis includes type checking, execution order, real-time requirements such as response times and worst-case execution times. This analysis helps in reducing late, costly and time-consuming testing efforts of, e.g., temporal errors. Furthermore, mathematical models and supporting tools provide formal evidence of fulfilling requirements.
  Using the Rubus analysis engines it is possible to timing analyze the system at various levels. For example, a single node is analyzed by calculating the response times of tasks and comparing them with corresponding deadlines. The analysis engines also support the analysis of end-to-end delays (such as Data Reaction and Age) for distributed systems (see Fig. 3). The analysis is based on

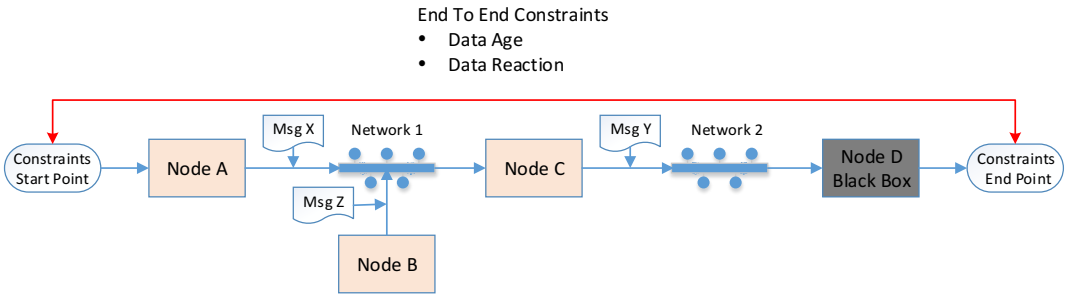

Fig. 3. System-level modeling and timing analysis of a distributed embedded system.

advanced data path analysis algorithms and supports multiple networks, black box nodes (whose internal software architectures are not available), message interference and redundant data paths [13], [12], [18].

- **Rubus Run-time Model (RRM) - Viewpoint of the run-time platform model:** The RCM design together with the RAM analysis is utilized to synthesize the code for the actual run-time platform. This automated synthesis prevents error-prone and costly integration errors. The run-time platform may be the Rubus Kernel or some other Real Time Operating System.

C. Certified Real-time Operating System

. The Rubus Real Time Operating System (RTOS) provides support for RCM in achieving an optimized real-time software system. The Rubus RTOS has been utilized in a wide variety of real-time applications. The main features of the Rubus RTOS are as follows:

- it supports the execution of time-triggered threads also called the Red threads;
- it supports the execution of interrupt-triggered threads also called the Green threads;
- it supports the execution of event-triggered threads also called the Blue threads;
- it supports communication between different types of threads;
- it supports static allocation of resources;
- and it supports scalability and portability.

The combination of a dynamic and static scheduling supported by the Rubus Kernel enables the design of optimized real-time software systems. The Rubus Kernel can be ported to various targets and development environments on customer's request and includes, amongst others, Freescale MPC-processors, Texas DSP, Infineons xc167-processors and various C-compiler environments such as Green Hills, WindRiver, Tasking, Microsoft VS and, GCC.

It is important to note that the Rubus Kernel has been approved as a certifiable ASIL D out of context element for real-time systems according to the automotive ISO 26262 standard (Road vehicle Functional Safety). There is an ongoing project to certify the Rubus Tool Suite according to this standard.

These concepts have proven to be effective in providing scalability from small to large real-time applications implemented by various organizations including Haldex, Borg Warner, Volvo Construction Equipment and BAE Systems (Hägglunds).

III. Related Models and Tools

There are several technologies and frameworks that support model- and component-based software development of embedded systems such as AADL [19], SCADE [20], MARTE [21], MAST [22], SysML, just to name a few. This paper targets the vehicular domain where the main focus is on EAST-ADL and EAST-ADL-like models for functional modeling and on AUTOSAR [23] and Rubus for execution modeling. Fig. 4 shows some of the models, approaches and tools that are used at four different abstraction levels defined by the EAST-ADL methodology. A detailed comparison of Rubus with several other models and tools is presented in [9]. It is important to note that with the implementation of some recent research results in Rubus Tool Suite, identified by ComSIS 2013 [13], ModComp 2014 [24], MASE 2015 [16], RTSCA 2015 [18], CBSE 2016 [25], ITNG 2016 [26] and SEAA 2016 [17] in Fig. 4, the tool suite now supports modeling and end-to-end timing analysis of distributed embedded systems at all the abstraction levels.

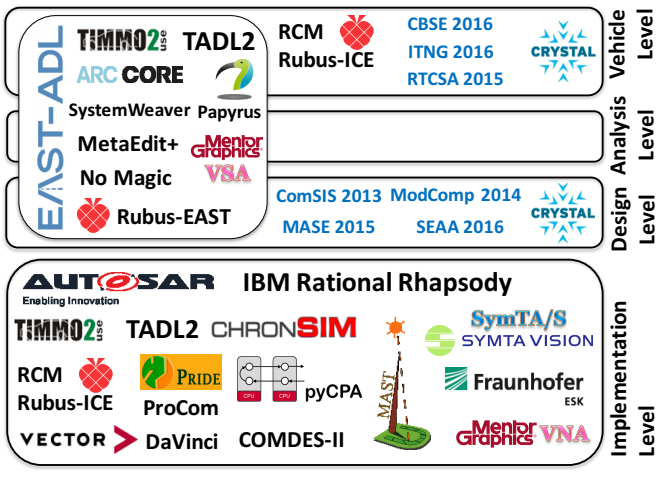

Fig. 4. Support for modeling and end-to-end timing analysis at various abstraction levels of EAST-ADL.

IV. Summary and Future Directions

The Rubus approach to providing both deterministic and non-deterministic service for vehicle systems has been proven by its usage as well as the certification of the Rubus Kernel according to the ISO 26262:2011 safety standard with respect to ASIL D. The evolution of Rubus from early experiences with deterministic solutions of automatic train control as well as the cooperation in the Vehicle Internal Architecture project that established the approach to mixing deterministic and non-deterministic services established the basis for the Rubus Kernel. The continued evolution both as a commercial product as well as a basis for research and development has stimulated cooperation between Arcticus Systems and Mälardalen University. This has included in provisioning of a Model-Driven Development tool suite that incorporates a variety of tools based upon research results.

In addition to the cooperation with Mälardalens University, Arcticus has participated in several European Research projects including TIMMO2USE[2], CRYSTAL[3], EMC2[4] and ASSUME[5].

Concerning the future, there are on going efforts to provide Rubus for Multi-core processors as well as developing a generic approach to model transformations in order to exchange information with other models and tool suites.

References

[1] Lawson, H., Wallin, S., Bryntse, B., and Friman, B., "Twenty years of safe train control in sweden," in *International Symposium and Workshop on Systems Engineering of Computer Based Systems, Washington, DC.*, 2001.

[2] ——, "Keynote address: Provisioning of safe train control in nordic countries," in *History of Nordic Computing Conference (HiNC3)*, 2008.

[3] Lawson, H., "Journey through the systems landscape," in *College Publications Systems Series, Volume 1, Kings College, London*, 2010.

[4] H. A. Hansson, H. W. Lawson, M. Strömberg, and S. Larsson, "Basement: A distributed real-time architecture for vehicle applications," *Real-Time Systems*, vol. 11, no. 3, pp. 223–244, 1996.

[5] 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.

[6] Eriksson, C., Lawson, H. and Lundbck, K-L., "A real-time kernel integrated with an off-line scheduler," in *IFAC/IFIP Workshop on Algorithms and Architecture for Real-Time Control*, 1997.

[7] Lawson, H. and Lundbck, K-L, "Provisioning of highly reliable real-time systems," in *History of Nordic Computing Conference (HiNC3)*, 2010.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] A. Bucaioni, A. Cicchetti, and M. Sjödin, "Towards a metamodel for the rubus component model," in *1st International Workshop on Model-Driven Engineering for Component-Based Software Systems*, September 2014.

[16] A. Bucaioni, A. Cicchetti, F. Ciccozzi, R. Eramo, S. Mubeen, and M. Sjödin, "Anticipating implementation-level timing analysis for driving design-level decisions in east-adl," in *International Workshop on Modelling in Automotive Software Engineering*, September 2015.

[17] A. Bucaioni, A. Cicchetti, F. Ciccozzi, S. Mubeen, M. Sjödin, and A. Pierantonio, "Handling uncertainty in automatically generated implementation models in the automotive domain," in *42nd Euromicro Conference series on Software Engineering and Advanced Applications*, September 2016.

[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] P. Feiler, B. Lewis, S. Vestal, and E. Colbert, "An Overview of the SAE Architecture Analysis & Design Language (AADL) Standard: A Basis for Model-Based Architecture-Driven Embedded Systems Engineering," in *Architecture Description Languages*, ser. The International Federation for Information Processing (IFIP). Springer US, 2005, vol. 176, pp. 3–15.

[20] SCADE Suite, http://www.esterel-technologies.com/products/scade-suite, accessed May, 2014.

[21] "The UML Profile for MARTE: Modeling and Analysis of Real-Time and Embedded Systems," January 2010. [Online]. Available: http://www.omgmarte.org/

[22] MAST–Modeling and Analysis Suite for Real-Time Applications, http://mast.unican.es/, accessed Mar. 2015.

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

[24] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Translating timing constraints during vehicular distributed embedded systems development," in *1st International Workshop on Model-Driven Engineering for Component-Based Software Systems*, September 2014.

[25] S. Mubeen, T. Nolte, M. Sjödin, J. Lundbäck, M. Gålnander, and K.-L. Lundbäck, "Modeling of Legacy Distributed Embedded Systems at Vehicle Abstraction Level," in *19th International Symposium on Component Based Software Engineering*, Apr. 2016.

[26] S. Mubeen, T. Nolte, J. Lundbäck, M. Gålnander, and K.-L. Lundbäck, "Refining Timing Requirements in Extended Models of Legacy Vehicular Embedded Systems Using Early End-to-end Timing Analysis," in *13th International Conference on Information Technology: New Generations (ITNG)*, Apr. 2016.

[2] https://itea3.org/project/timmo-2-use.html

[3] http://www.crystal-artemis.eu

[4] http://www.artemis-emc2.eu/

[5] https://itea3.org/project/assume.html