Arcticus Systems

Arcticus – The Provider of Rubus®

WHITE PAPER

2016-01-19

Kurt-Lennart Lundbäck, John Lundbäck, Harold “Bud” Lawson,
Mikael Sjödin and Saad Mubeen

1 Introduction

Arcticus Systems AB delivers and supports state-of-the-art safety critical embedded real-time system products and services. The products are called **Rubus** (the name of a red arctic berry and a registered trademark of Arcticus AB).

The Rubus suite provides a collection of methods and tools for the Model Driven Development of real-time systems. Rubus enables the designer to graphically describe systems as interconnected components. These interconnected components (following a hardware paradigm called Software Circuits) define the structure of the application system that can be analyzed and synthesized entirely within the Rubus environment.

Rubus integrates modelling, timing analysis, code synthesis (including a dedicated Real Time Operating System (the Rubus Kernel)) and simulation of the combined application software and the Rubus Kernel.

A unique property of the Rubus suite is the ability (at pre-runtime) to analyze and verify both time driven deterministic (triggered) and event driven non-deterministic (triggered) application tasks with respect to timing properties and resource utilization.

Another important property of the Rubus suite is the ability to develop the real-time application software independent of the target processor and execution platform.

The Arcticus strategy is to work in close cooperation with customers by providing our products and expertise to assure the successful development and support of their embedded real-time applications. Further, to be a leading developer of state-of-the-art real-time products based upon cooperation with leading universities, especially with Mälardalen University, and by active participation in national and international research programs.

In this white paper, the evolution of Rubus, the driving concepts used in Rubus, a description of the Rubus products, the research contributions from Mälardalen University, the Arcticus approach to customer projects and on-going research and development projects are presented.

2 The Evolution of Rubus

Arcticus Systems AB was established as a Swedish corporation in 1985. The first product (called O'Tool) provided a real-time kernel for facilitating the implementation of event-driven real-time systems. The O'Tool product had a significant international market including such important customers as Rank Xerox (deployed in copying machines) and ABB (in power control equipment).

The Rubus Kernel was first introduced for industrial use in 1996. The concepts utilized in Rubus evolved from previous experiences with O'Tool as well as Arcticus 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 1970's 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 Standard Radio on-board system is now owned and further developed by Ansaldo of Italy and is often supported by the original developers at Teknogram (now owned by ÅF) in Borlänge, Sweden. Ansaldo has used the software architecture for developing other products including the provisioning of the ATP (P for Protection) system used on the New Jersey Transit. Since its inception and updating in the 1990's for X2000 trains, this architecture has remained stable and delivered the reliability required for the on-board ATC function. The results of this product development have been reported in (Lawson, et.al, 2001), (Lawson, 2008) and (Lawson, 2010).

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. Blue application tasks are executed within residual time in the time slots. 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 (Hansson, et.al, 1996a and 1996b).

Arcticus established an agreement with Mälardalens 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. This work was reported at an IFAC/IFIP conference (Eriksson, et.al, 1997). This effort formed the basis for the eventual Model Driven Development 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 TorqTransfer Systems). In addition to Arcticus, Lawson Konsult AB, Uppsala University and Mälardalens Högskola participated in this project. After several prototype iterations the first LSCD was delivered to Volkswagen. Since then it has evolved into a significant product that it utilized in most all four wheel drive vehicles in the world. The system is now owned by Borg-Warner.

In 1997, both Volvo Construction Equipment 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 Model Driven Development products (Rubus Tool Suite).

Volvo Construction Equipment (VCE) has deployed the Rubus Kernel in several of its products. Most recently, it has been decided that the Rubus Kernel should be made certifiable according to the international standard ISO 26262 (Road Vehicles – Functional Safety). As a part of this effort, Arcticus has developed its QMS (Quality Management System) based upon the now harmonized ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207 on Systems, respective Software Life Cycle Processes.

Arcticus has participated, with several of their customers and research partners, in KK-Stiftelsen (The Knowledge Foundation) sponsored projects, including MultEx, EEMDEF and FEMVA. Arcticus has also participated and continues to participate in a number of EUROPEAN-research projects including TIMMO2USE, CRYSTAL and EMC2. All of these projects have contributed to the progress in providing the state-of-the-art Rubus product suite.

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 (Lawson and Lundbäck, 2010). This publication also includes an overview of the Arcticus products and their customers.

3 Driving Concepts

Based upon the evolution of Arcticus products, a few central concepts have evolved that drive the product related thinking as follows:

3.1 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. The model of a software circuit is shown in Figure 1.

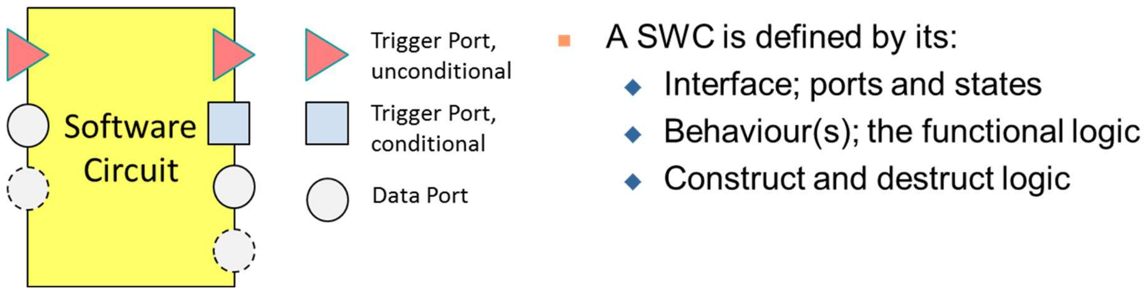
Software Circuit
Trigger Port, unconditional
Trigger Port, conditional
Data Port
A SWC is defined by its:
Interface; ports and states
Behaviour(s); the functional logic
Construct and destruct logic

Figure 1: Model of a Software Circuit.

The Construct and Destruct logic are terms from Object Oriented languages that describe activation, respectively, deactivation of the circuit.

3.2 Time Triggered, Event Triggered, and Interrupt Execution

Software Circuit execution is triggered based on these three categories. The ability to mix both deterministic (time triggered) and non-deterministic (event triggered) as well as treating interrupt execution is a unique aspect of the Rubus product suite.

3.3 Model Driven Development

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 Figure 2.

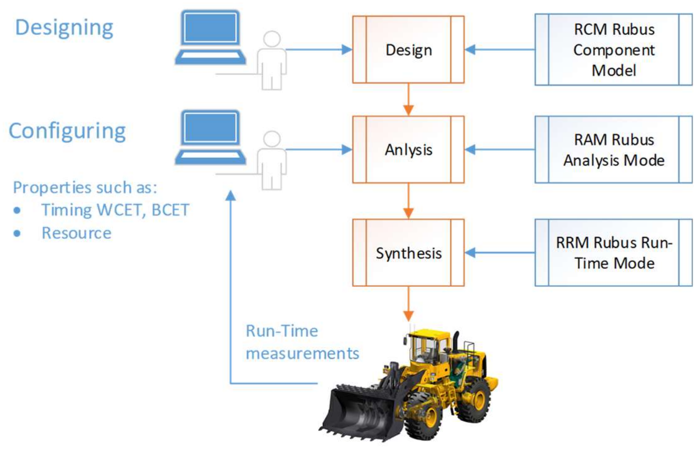

Figure 2: Rubus Conceptual Models.

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

- **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.
- **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.
- **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.

These concepts have proven to be effective in providing scalability from small to large real-time applications implemented by various organizations.

4 Rubus Products

The main products; namely Rubus Tool Suite and Rubus RTOS that Arcticus delivers to its customers are portrayed in Figure 3. It is important to note that the real-time application developed using Rubus Tool Suite can be executed on a variety of real-time platforms, that is, various hardware and various real-time operating systems.

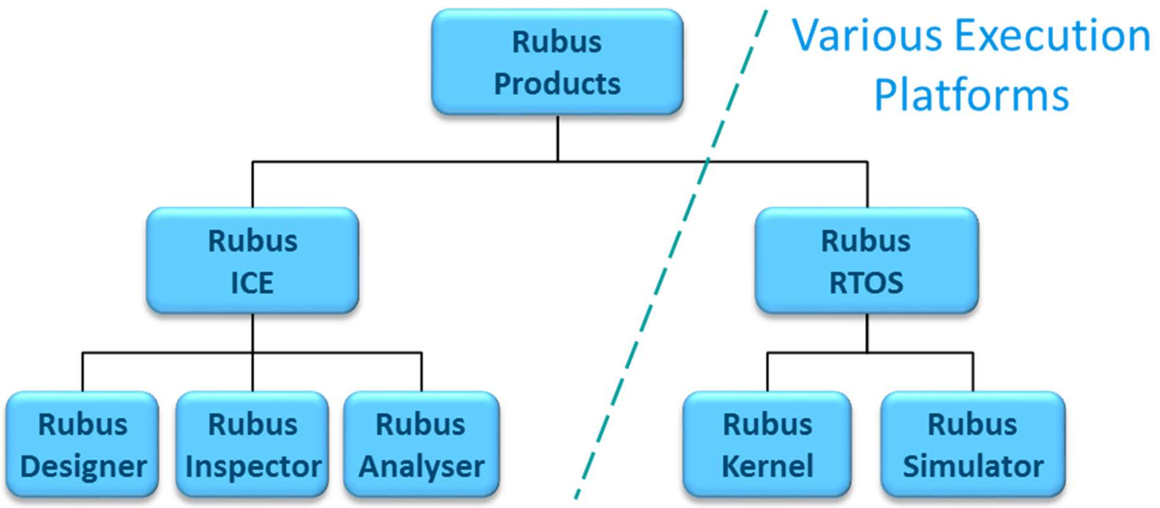

Figure 3: The Rubus Product Suite.

4.1 The Rubus Tool Suite Product

The Rubus Tool Suite (Integrated Component Model Development Environment) that is utilized by customers in developing, simulating and implementing time critical and non-time critical applications.

Confidential/proprietary information of Arcticus Systems AB is contained herein and may not be disclosed, displayed, used reproduced or copied without prior written consent. Failure to comply with this notice may result in liability for costs, damages or losses.
© 2018, Arcticus Systems AB. All rights reserved. Rubus® is a registered trademark of Arcticus Systems. 5

Rubus Tool Suite provides for the implementation of the Model Driven Development concept by providing the following properties:

- Raises the level of abstraction thus addressing the increasing complexity problem for embedded systems software.
- Formal and early reasoning. The system architecture/design 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 real time model and the program code. The user works on a platform independent model, then selects the specific target platform, and the Rubus Tool Suite tool generates the framework code. Productivity is increased since the auto-generation automates code generation that is often error prone.
- Traceability. The system architecture/design documentation is kept up to date with the implementation resulting in traceability from design to implementation and vice versa.

Rubus Designer is used to interactively describe and analyse the application component structure as Software Circuits in forming a Rubus Component Model (RCM), as portrayed in Figure 2. The user defines the input and output ports of components and connects the Software Circuits in a manner similar to hardware diagrams as illustrated in Figure 5.

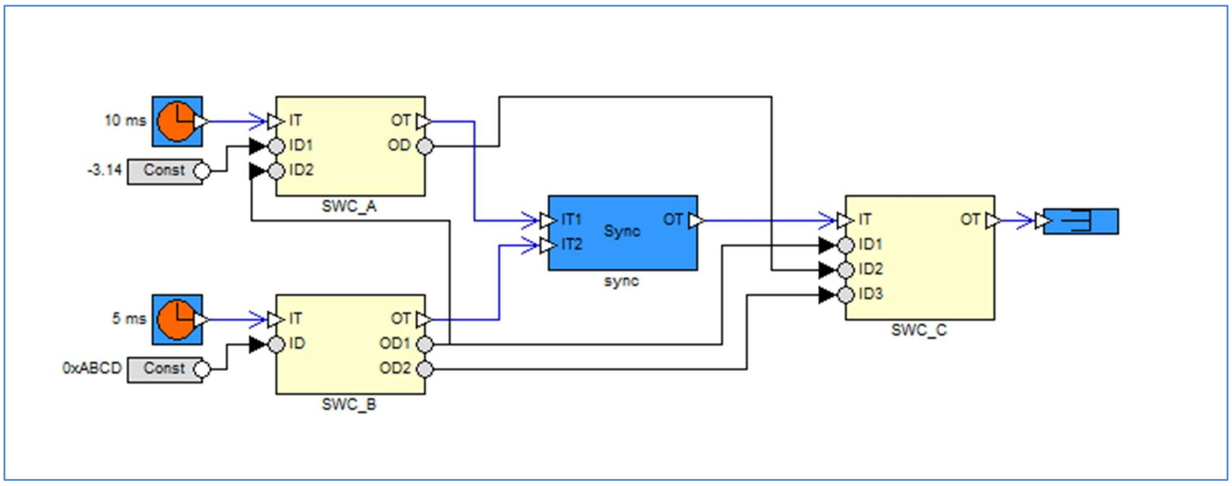

Figure 5: Combining Software Circuits

The Rubus Analysis Model (RAM) was developed to be incorporated in the Rubus Designer, in cooperation with research partners at Mälardalens University and is provided as a part of the Rubus Designer tool. This initially provided the basis for an off-line scheduler as described in (Eriksson, et al, 1997). Further development included response-time analysis of tasks with offsets (Mäki-Turja, et.al, 2004), shared-stack analysis (Hänninen, et.al, 2008) and response-time analysis of Controller Area Network (CAN) and its higher-level protocols (Mubeen, et.al, 2015), as well as distributed end-to-end response time and path delay analysis (Mubeen, et.al, 2013).

Using the timing analysis support in Rubus Designer, it is possible to analyze a single node by calculating the response times of tasks and comparing them with corresponding deadlines. Rubus Designer also supports the analysis of end-to-end delays (such as Data Reaction and Age) for distributed systems (see Figure 6 below). The analysis is based on 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. The internal execution models of nodes are taken into account if available. It is possible to analyse network communication based on Controller Area Network (CAN) or Ethernet by calculating the response times of messages. This distributed analysis is a unique property provided by the Rubus Designer.

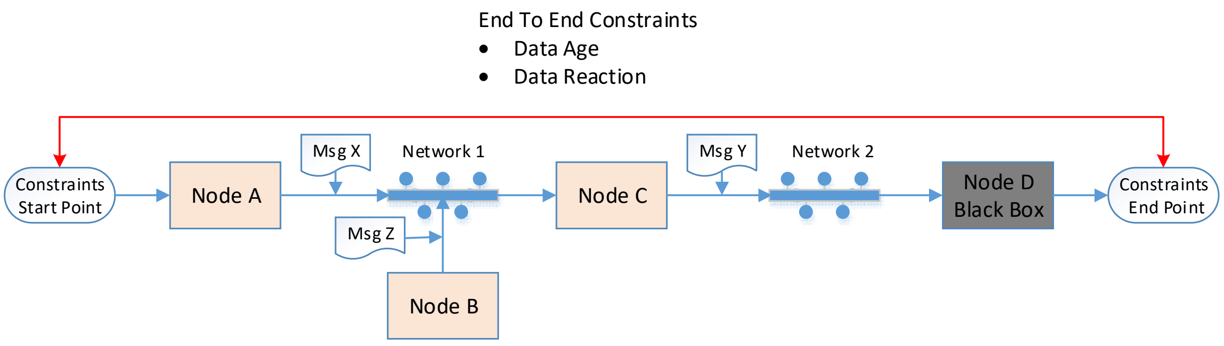

Figure 6: End-to-End Network Analysis

Rubus Inspector provides for platform independent formal and semi-formal "Model in the Loop" testing environment used to test the Software Circuits of the Rubus Component Model (RCM). It provides for unit testing as well as for sub-system or complete system testing.   Test inputs are provided by the user however, it is also possible to generate tests that utilize Simulink and Matlab environments.

Rubus Analyser provides for the user-friendly presentation of off-line and on-line information about the real time 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.

4.2 The Rubus RTOS Product

While the Rubus Component Model developed utilizing Rubus Tool Suite is platform independent, Arcticus supplies the Rubus RTOS that has been utilized in a wide variety of real-time applications.

Rubus Kernel that is integrated with the application software and further integrated into the customers products.

Rubus Simulator that provides for testing and verifying the composite of the Rubus Kernel and application software.

Rubus Kernel provides support for the Rubus Component Model (RCM) in achieving an optimal real-time software system. The main features of the Rubus Kernel are:

- Support the execution of Time-Triggered Red threads.
- Support the execution of Interrupt-Triggered Green threads.
- Support execution of Event-Triggered Blue threads.
- Support communication between different types of threads.
- Support statically allocated resources.
- Support scalability and portability.
- Support the instrumentation of runtime analysis.

The combination of a dynamic and static scheduling supported by the Rubus Kernel enables the design of optimal real-time software systems.

The Rubus Kernel can be ported to various targets and development environments on customer 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.

5 Contributions from Mälardalens University

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-suite. 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 is shown in Figure 7.

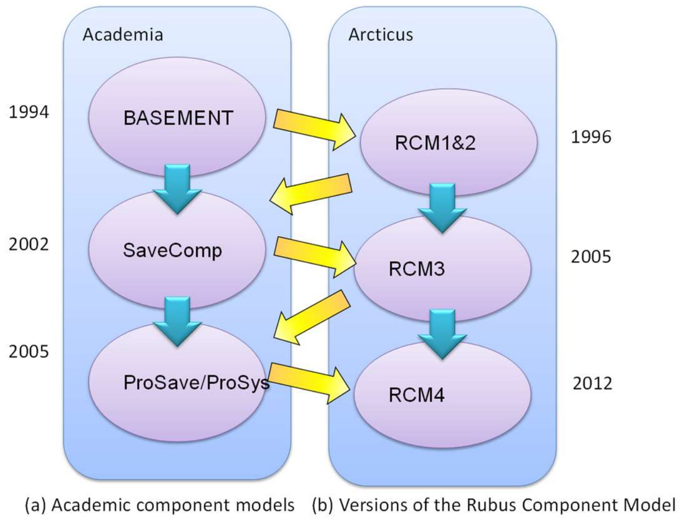

Figure 7: Contributions to the Rubus Component Model.

As mentioned above, Rubus was initially developed from the joint academic and industrial project VIA project BASEMENT and its conceptual component-model. In development of the first Rubus tools, MDH researchers Christer Norström (former Eriksson) and Kristian Sandström developed the first scheduling algorithms for the Red threads (see Eriksson, et al, 1997).

Later an SSF[1] frame-project, SAVE, coordinated by Hans Hansson at MDH with partners from KTH, Linköping University, and Uppsala University based the component-model SaveComp on the BASEMENT-concept and the experiences from early Rubus models. In a technology-transfer project MULTEX, funded by KKS[2], MDH's researchers Mikael Sjödin and Jukka Mäki-Turja worked with Arcticus to define the next version of the Rubus Component Model (RCM) using concepts from SaveComp. This resulted in the first Rubus-version where components could be used in Green, Red, and Blue threads.

In 2006 SSF granted another frame-project, PROGRESS, to MDH (also led by Hans Hansson) to continue work on software engineering using component-based technologies. Again, results from PROGRESS, specifically the ProComp component model, inspired further development of the Rubus Component Model and a new technology-transfer project, EEMDEF, was granted to Jukka Mäki-Turja in 2009 by the KKS. The EEMDEF project resulted in the extension of RCM and Rubus Tool Suite to support the modeling and development of distributed embedded systems (Mubeen, et.al, 2014a).

Another KKS-funded project FEMMVA resulted in the extension of timing analysis framework of Rubus Tool Suite to support the specification, analysis and validation of end-to-end path delay constraints namely age and reaction. Further, the results of the project provided foundations for the translation of timing constraints and design-level models from EAST-ADL (an architecture description language in the automotive domain) to RCM (Mubeen, et.al, 2014b).

Since 2012, and the finalization for Rubus Component Model version 4, Arcticus have further intensified the academic cooperation with MDH and with participation in several larger European projects. Many projects are a result of the continued cooperation with MDH's Mikael Sjödin and his research group Model-Based Engineering of Embedded Systems (MBESS) that include senior researchers Jukka Mäki-Turja and Saad Mubeen. Both Jukka and Saad have during the last years been employed by Arcticus as industrial research-engineers and industrial PhD-students respectively. Both these industrial positions have been funded by VR[3]. Other industrial PhD-students from MBEES that have been supported by Arcticus include Kaj Hänninen and (ongoing) Alessio Bucaioni. Finally, a Worst Case Execution Time analysis project done in cooperation with Björn Lisper provided an important step toward adding WCET analysis of the C-source code as a complement to measurement on target processors.

6 Customer Projects

Each agreement to deliver and support a Rubus Software System Product results in the establishment of a Project that is based upon the Customer Need and an Agreement as indicated in Figure 8.

[1] Stiftelsen för Strategisk Forskning, Swedish foundation for strategic research
[2] KK-stiftelsen, the Knowledge foundation
[3] Vetenskapsrådet, the Swedish research council

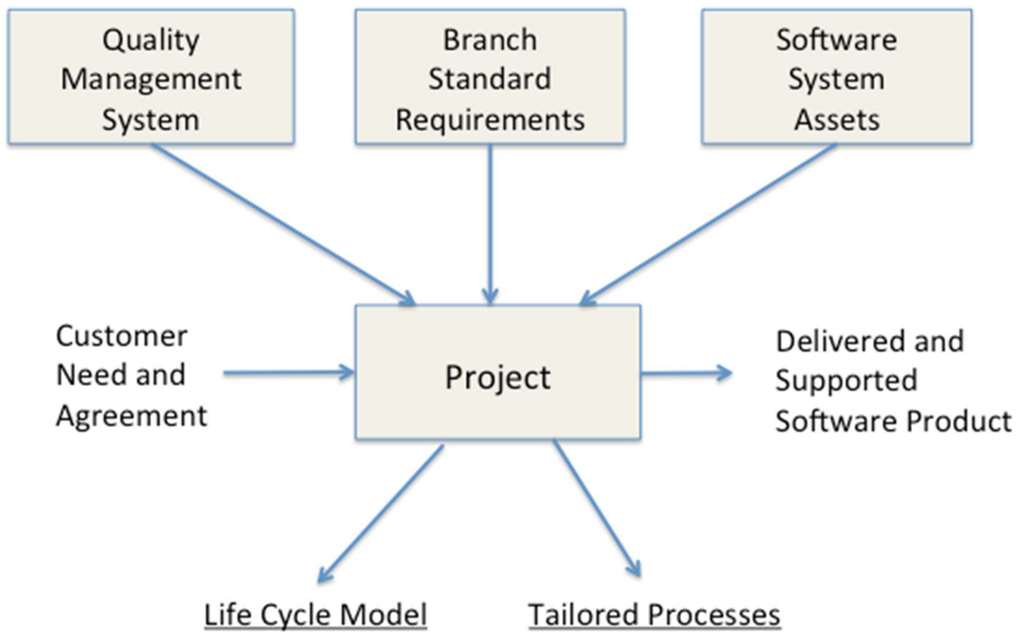
Quality Management System
Branch Standard Requirements
Software System Assets
Customer Need and Agreement
Project
Delivered and Supported Software Product
Life Cycle Model
Tailored Processes

Figure 8: Project Establishment.

In establishing a project, the Arcticus Quality Management System is used as the basis and then further tailored to take account of customer requirements in respect to branch standards. This results in an appropriate life cycle model and the processes to be followed in the project. The project often makes use of the existing Rubus Software System Assets. Arcticus works closely with the customer during the project in a manner similar to an Integrated Project Team (IPT).

6.1 Safety Standard ISO 26262

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.

7 On-Going Research and Development

7.1 DPAC

The objective of DPAC (2015-2023) acronym for "Dependable Platforms for Autonomous systems and Control" is to enhance and establish a strong research profile at Mälardalens University (MDH), targeting dependable platforms for autonomous systems[2] and control. The new profile will strengthen the interdisciplinary collaboration from five different research areas within the Embedded Systems (ES) environment at MDH, thus, allowing for a holistic view on dependable platforms, and making new collaborations and positive synergy effects possible.

Within DPAC, Arcticus participates in the project "Predictability and dependability in parallel architectures". This project includes a set of work packages that will contribute towards improved software support for reconfigurable and dependable use of parallel architectures, ranging from contemporary multi-parallel multicores to future hyper-parallel HSA chips.

7.2 EMC2

Arcticus participates in the EU ARTEMIS research project EMC² (2014-2017) acronym for “Embedded Multi-Core systems for mixed criticality applications in dynamic and changeable real-time environments”. The mixed criticality captures both static (pre-run-time configuration) as well as the dynamic (run-time behaviour) criticality. The objective of the EMC² project is to foster these changes through an innovative and sustainable service-oriented architecture approach for mixed criticality.

There is a clear trend towards mixed critical systems resulting in the need for certified run-time systems as well as development models and tools to support the development of such systems. The software complexity must be addressed by utilizing safe and sound run-time systems and models and tools.

The project has resulted in the evaluation of the Rubus component model in the context of mixed criticality applications to be deployed on multi/many-core architectures. Strategies for partitioning and assignment are being developed along with supporting methods for resource management and scheduling.

As a result of this project, Arcticus is developing a hypervisor solution for Model Driven Development to facilitate the implementation of various types of critical (safety critical) and non-critical applications.

7.3 CRYSTAL

Arcticus participates in the EU ARTEMIS project CRYSTAL (2013-2016) acronym for CRitical SYSTem Enginieering AcceLeration.

CRYSTAL aims at fostering Europe’s leading edge position in embedded systems engineering in particular regarding quality and cost effectiveness of safety-critical embedded systems and architecture platforms.

There is an ever increasing product variability and the need to speed up time-to-market and reduce development time. The tighter the loop between design and design evaluation (analysis) the better the overall design.

In this project Arcticus focuses upon model driven pre-run-time analysis in order to minimize post-built system testing. Thus providing the software engineer with one holistic analysis framework.

Arcticus has extended the analysis algorithms (RAM) in the Rubus CM by enabling the seamless combination of functional constraints and requirements with target environment constraints including resources based on the latest EAST-ADL specification and the TADL2 timing model. The focus is placed upon end-to-end analysis of data communication in distributed real-time systems as well as ability to import and export EAST-ADL model files.

7.4 ASSUME

Arcticus also participates in the EU ITEA3 project ASSUME (2015-2018) acronym for Affordable Safe & Secure Mobility Evolution.

The ASSUME project aims at providing a seamless engineering methodology for delivering trustworthy new mobility assistance functions on multi and many core architectures. The problem is addressed on the constructive and on the analysis side. For efficient construction and synthesis of embedded systems, the project provides new tools, standards, a methodology and interoperability solutions to cover most of the challenges by design. The project includes five countries with a total effort of 202 person years.

Arcticus has developed the Rubus environment for Model Driven, component-based development of embedded real-time software. The supported development process includes a strict discipline for pre-run-time analysis of real-time properties. This analysis requires WCET estimates for tasks. Currently these have to be provided manually.

To allow for an automation of this provision, Arcticus will define and implement a plugin interface for WCET analysis tools. This will be done in cooperation with Mälardalens University in Sweden, and the interface will be validated using SWEET tool.

8 Acknowledgements

Arcticus thanks all of their customers and research and development partners for their active participation in making Rubus the state-of-the-art products that they have become. In particular, KK-Stiftelsen, Vinnova and SSF that have supported various projects. The significant contributions of research partners at Mälardalen Univeristy is hereby acknowledged via the contributions of Hans Hansson, Christer Norström, Kristian Sandström, Mikael Sjödin, Jukka Mäki-Turja, Kaj Hänninnen, Saad Mubeen and Björn Lisper. Harold ”Bud” Lawson has made important contributions to the evolution of the Rubus concepts as well as developing the Arcticus Quality Management System and participating as the Safety Manager in the ISO 26262 certification of the Rubus Kernel.

9 References

Eriksson, C., Lawson, H. and Lundbäck, K-L. (1997) A Real-Time Kernel Integrated with an Off-Line Scheduler, IFAC/IFIP Workshop on Algorithms and Architecture for Real-Time Control.

Hansson, H., Lawson, H., Strömberg, M., and Larsson, S. (1996a) BASEMENT: A Distributed Real-Time Architecture for Vehicle Applications, Proceedings of the IEEE Real-Time Applications Symposium, Chicago, IL, May 1995. Also appearing in Real Time Systems, The International Journal of Time-Critical Computing Systems, Vol. 11. No. 3,

Hansson, H., Lawson, H., Bridal, O., Ericsson, C., Larsson, S., Lön, H., and Strömberg, M, (1996b) BASEMENT: An Architecture and Methodology for Distributed Automotive Real-Time Systems, IEEE Transactions on Computers, Vol. 46. No. 9

Bohlin, M., Hänninen, K., Mäki-Turja, J., Carlson, J. and Sjödin, M. (2008) Bounding Shared-Stack Usage in Systems with Offsets and Precedences, Euromicro Conference on Real Time Systems, July, 2008.

Lawson, H., Wallin, S., Bryntse, B., and Friman, B. (2001) Twenty Years of Safe Train Control in Sweden, Proceedings of the International Symposium and Workshop on Systems Engineering of Computer Based Systems, Washington, DC.

Lawson, H, (2008) Provisioning of Safe Train Control in Nordic Countries, Keynote address appearing in the Proceedings of HiNC2, History of Nordic Computing

Lawson, H. (2010) A Journey Through the Systems Landscape, College Publications Systems Series, Volume 1, Kings College, London.

Lawson, H. and Lundbäck, K-L (2010) Provisioning of Highly Reliable Real-Time Systems, appearing in the Proceedings of HiNC3, History of Nordic Computing

Mäki-Turja, J. and Sjödin, M. (2004) Tighter Response-Times for Tasks with Offsets, International Conference on Real-time and Embedded Computing Systems and Applications Conference, August, 2004.

Mubeen, S., Mäki-Turja, J. and Sjödin, M. (2013) Support for End-to-End Response-Time and Delay Analysis in the Industrial Tool Suite: Issues, Experiences and a Case Study, Journal of Computer Science and Information Systems, Vol. 10, No. 1.

Mubeen, S., Mäki-Turja, J. and Sjödin, M. (2014a) Communications-Oriented Development of Component- Based Vehicular Distributed Real-Time Embedded Systems, Journal of Systems Architecture, Vol. 60, No. 2, 2014, Elsevier.

Mubeen, S., Mäki-Turja, J. and Sjödin, M. (2014b) Component-Based Vehicular Distributed Embedded Systems: End-to-end Timing Models Extraction at Various Abstraction Levels, MRTC Report, Mälardalen University Sweden, 2014, MDH-MRTC-285/2014-1-SE.

Mubeen, S., Mäki-Turja, J. and Sjödin, M. (2015) Integrating mixed transmission and practical limitations with the worst-case response-time analysis for Controller Area Network, Journal of Systems and Software, Vol. 99, 2015, Elsevier.