Timing Verification of Component-based Vehicle Software with Rubus-ICE: End-user's Experience

Saad Mubeen[1], Mattias Gålnander[2], Alessio Bucaioni[2], John Lundbäck[2], Kurt-Lennart Lundbäck[2]
[1]Mälardalen University, Sweden
[2]Arcticus Systems, Sweden
[1]saad.mubeen@mdh.se
[2]{mattias.galnander, alessio.bucaioni, john.lundback, kurt.lundback}@arcticus-systems.com

ABSTRACT

This paper discusses an end-user's experiences of utilizing timing analysis tools to verify predictability of distributed embedded systems in the vehicle industry. The analysis tools are plug-ins for an industrial tool suite, namely Rubus-ICE, that is based on the principles of model-based engineering (MBE) and component-based software engineering (CBSE). These plug-ins implement various state-of-the-art timing analyses including response-time analysis and end-to-end data-path analysis. The experiences discussed in this paper provide a useful feedback in terms of usability and validity of assumptions to the tools provider as well as to the academia.

CCS CONCEPTS

• **Computer systems organization → Embedded software** ;

KEYWORDS

Automotive software, End-to-end timing analysis, CBSE, MBE.

1 INTRODUCTION

The continuous increase in size and complexity of vehicle software (SW) makes its development more and more challenging. The component models and tools that are based on MBE [5] and CBSE [3] are proving effective in dealing with the SW complexity, e.g., AUTOSAR [2], Rubus Component Model (RCM) [4] and Rubus-ICE (Integrated Component development Environment) [8]. In addition to the SW complexity, many vehicular embedded systems are required to be predictable, i.e., all actions by the systems are performed in a timely manner such that all timing requirements are satisfied [9]. Hence, pre-runtime verification of predictable timing behavior is another challenge that is faced by the system providers. One way to deal with this challenge is to integrate schedulability analysis [1, 8] with the SW development tools to allow timing verification at higher abstraction levels, e.g., at the level where the SW architecture is developed without any knowledge of low-level implementation details. Within the context of the above discussion, this paper considers a component model and its tool chain, namely RCM and Rubus-ICE respectively. The Rubus concepts, models and tools have been developed and evolved in a close collaboration between academia and industry over the last three decades [7]. The models and tools have been used for model- and component-based SW development of predictable, timing analyzable and synthesizable embedded systems in the vehicle industry for over 20 years.

SQUADE'18:, May 28, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5737-1/18/05.
https://doi.org/10.1145/3194095.3194103

Various features supported by Rubus are depicted in Fig. 1. Some of the notable models and tools supported by Rubus are as follows.

- *Component Model and Designer Tool:* The Rubus designer, based on RCM, supports graphical modeling of SW architecture by interconnecting reusable SW components (SWCs). It also allows for modeling and specifying timing information on the architecture.
- *Analysis Framework:* The Rubus analysis framework support model-based analysis of the SW architecture at various abstraction levels. It helps in mitigating late, costly and time-consuming testing efforts. The analyses include response-time and end-to-end data-path analyses of distributed embedded systems [8].
- *Automatic Code Generator:* This tool supports automatic code generation of run-time environment. Behavioral code can be generated, thanks to the integration with Simulink.
- *Rubus Real-time Operating System (RTOS):* A system developed using RCM can be executed on a variety of processors and RTOSs. Rubus provides a support for an ISO26262 compliant RTOS, which allows both time- and event-triggered execution.
- *Simulation and Testing Tools:* These tools support model-in-the-loop testing at various levels (from unit testing to system testing) and simulation in various host environments.

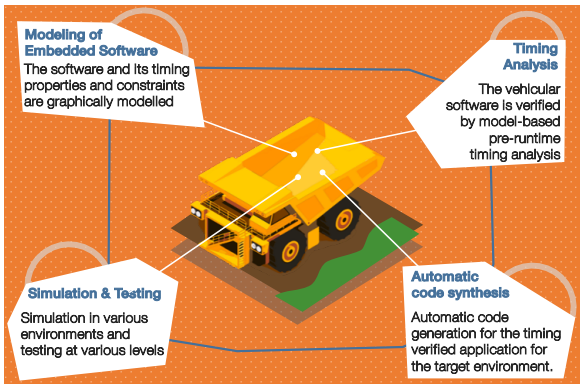

Figure 1: Various features supported by Rubus-ICE.

2 END-USER'S EXPERIENCES

This section provides the experiences of an end-user from the construction vehicles domain regarding the usability, correctness, reliability and effectiveness of Rubus tools.

Usability. Ease of use of the SW development tools is an important requirement from the end-user. RCM and Rubus tools have evolved as a result of long-term collaboration among academia, tools developer and end-users. Usability is one of the properties that has improved quite significantly as a result of the collaboration.

The average end-user of Rubus-ICE is only aware of the component model, its modeling concepts and features of the designer tool. Further, the end-user is abstracted from algorithms and implementation details of the analysis plug-ins to maximize the usability and to minimize/mitigate the problems faced by inexperienced engineers.

Early detection of timing issues. The end-user considers the pre-runtime formal analysis (integrated to the component model) to be the real strength as it allows to detect timing issues (e.g., response times and delays not meeting the specified timing requirements). These issues can not be detected with testing as the exhaustive testing is not viable with respect to cost and time to market.

Speed of timing analysis. The speed of finding the analysis results carries high importance for the end-user. Due to iterative nature, many timing analyses can take large times (e.g., hours) to provide the results [8], which may not be feasible in practice. Generally, there are three different ways to deal with this challenge. First, make the analysis faster by using approximate functions instead of exact functions, e.g., [6]. This method can provide the analysis results faster but at the cost of over-estimation or pessimism. Second, make the analysis faster by using detailed knowledge of the RTOS to avoid analysis of the cases that cannot happen, e.g., considering the effect of exact schedule in the analysis. Third, optimize the implementation of the analysis. Note that Rubus-ICE uses the second and third methods to speed up the time to perform the analysis.

Academic assumptions vs industrial settings. The timing analysis implemented in the tools [8] considers worst-case assumptions and accounts for worst-case scenarios. As a result, the calculated response times and delays can be pessimistic (or over-approximated). One could argue that the worst-case scenario considered in the analysis might not occur in reality or may occur very rarely. Thus, a system designed based on the worst-case timing analysis can greatly under-utilize the system resources. We argue, based on the experiences of the end-users of the tools, that the worst-case scenario can be observed in reality more often than it is perceived. Consider an example of a two-node distributed embedded system shown in Fig. 2. Each node contains one task (a task corresponds to a SWC at runtime), which is activated independently with a period of 10 time units. The nodes are connected by a Controller Area Network (CAN), which is one of most widely used in-vehicle networks. Task τ_1 in Node1 sends a message m_1 to τ_2 in Node2. One of the worst-case assumptions used by the state-of-the art timing analysis is concerned with the time at which the receiving task (τ_2) is activated to read the received message (m_1). Consider three different cases in Fig. 2. In case (a), τ_1 and τ_2 are activated at the same time in their respective nodes. m_1 arrives at Node2 at time 5; however, τ_2 is able to read m_1 only upon its activation at time 10. The Age delay from τ_1 to τ_2 is identified in Fig. 2. In case (b), τ_2 is activated as soon as m_1 arrives at node2. This results in a shorter Age delay compared to case (a). In case (c), it is assumed that τ_2 is activated just before the arrival of m_1 at Node2. Hence, the first instance of τ_2 just misses the read access of m_1, which is later read by the second instance of τ_2, resulting in the largest Age delay among all cases. Case (c) corresponds to one of the worst-case assumptions used by the timing analysis tools. The end-user experienced that the worst-case scenario and near worst-case scenarios can often occur due to event-driven network communication, nodes lacking synchronization and clock drifts in nodes. Hence, the worst-case assumptions must be considered by the analysis tools for timing verification of safety-critical vehicular embedded, even at the cost of under-utilization of system resources.

Interoperability. There is a plethora of modeling, analysis and test tools that are used during the SW development of vehicular embedded systems. The end-user often demands interoperability among various tools provided by different vendors. It is challenging to provide a seamless tool chain due to incompatible abstractions, semantic gaps, and non-standard exchange formats. The tools integration often comes at the cost of relaxed semantics and restrictions.

Guidelines for good design. The main challenge for the inexperienced end-users is how to develop good SW that satisfies the specified timing requirements. The end-user can develop bad models by misusing even efficient tools. Providing such guidelines (influenced by the timing analysis tools) to the end-user is a difficult challenge faced by the tool providers and academia. Solving this challenge can be very helpful and cost effective for the end-users.

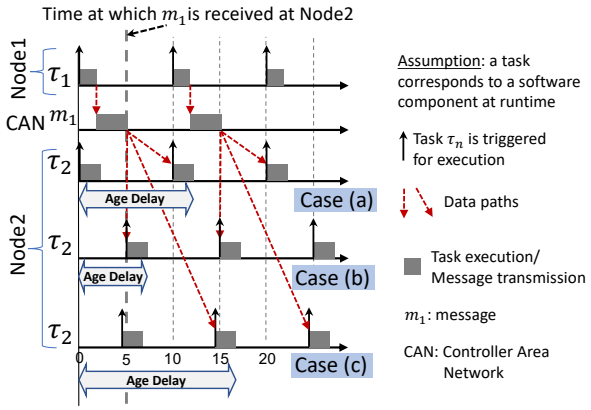

Figure 2: Effect of different assumptions on delays.

3 SUMMARY AND CONCLUSION

This paper discussed the experiences, needs and challenges that are faced by the end-user of the Rubus Component Model (RCM) and its tool chain Rubus-ICE, which are used for the SW development of predictable vehcular embedded systems. The paper also provided an insight into the importance of various aspects in the industry that include usability, early detection of timing errors, speed of timing analysis tools, relevance of the worst-case assumptions behind the timing analysis tools, interoperability and guidelines to design good models of predictable vehicular embedded systems.

Acknowledgment The work in this paper is partly supported by the Swedish Knowledge Foundation (KKS) via the PreVeiw project.

REFERENCES

[1] N.C. Audsley, A. Burns, R.I. Davis, K. Tindell, and A.J. Wellings. 1995. Fixed priority pre-emptive scheduling: an historic perspective. *Real-Time Sys.* 8, 2/3 (1995).
[2] AUTOSAR Tech. Overview. 2013. rel.4.1, rev.2, ver.1.1.0., http://autosar.org. (2013).
[3] Ivica Crnkovic and Magnus Larsson. 2002. *Building Reliable Component-Based Software Systems.* Artech House, Inc., Norwood, MA, USA.
[4] K. Hänninen et.al. 2008. The Rubus Component Model for Resource Constrained Real-Time Systems. In *IEEE SIES.*
[5] Thomas A. Henzinger and Joseph Sifakis. 2006. The Embedded Systems Design Challenge. In *14th International Symposium on Formal Methods.* Springer, 1–15.
[6] J. Maki-Turja and M. Nolin. 2005. Fast and tight response-times for tasks with offsets. In *17th Euromicro Conference on Real-Time Systems.* 127–136.
[7] S. Mubeen, H. Lawson, J. Lundbäck, M. Gålnander, and K. L. Lundbäck. 2017. Provisioning of Predictable Embedded Software in the Vehicle Industry: The Rubus Approach. In *IEEE/ACM 4th International Workshop on Software Engineering Research and Industrial Practice.* 3–9. https://doi.org/10.1109/SER-IP.2017..1
[8] Saad Mubeen, Jukka Mäki-Turja, and Mikael Sjödin. 2013. 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* 10, 1 (2013).
[9] John A. Stankovic and Krithi Ramamritham. 1990. What is predictability for real-time systems? *Real-Time Systems* 2, 4 (01 Nov 1990), 247–254.