Bringing MoVES Towards Consolidated Electrical/Electronic Automotive Architectures

Alessio Bucaioni, Saad Mubeen
Mälardalen University, Sweden
{alessio.bucaioni, saad.mubeen}@mdh.se

I. Introduction

During the last decades, automotive software systems have been evolving at a staggering pace. Through the years, several model-driven methodologies have been introduced for the development of automotive software systems. As verification of timing predictability became a pivotal task for the homologation and safety certification of these systems, in our previous work, we have introduced MoVES, a model-driven methodology for automotive software supporting the development and architectural exploration of system designs with temporal awareness [1]. To this end, MoVES exploits two industrial automotive-specific modelling languages, EAST-ADL [2] and Rubus Component Model (RCM) [3], and a set of six model transformations.

Currently, the automotive industry is exploring new mobility solutions with future vehicles being adaptive, autonomous and connected [4]. However, these solutions introduce new challenges both at software and hardware level, which in turn affect timing predictability analysis of the software architectures. Upcoming automotive software systems will demand for computational power and high bandwidth for on-board communication well beyond the capacity of most of the contemporary automotive Electronic Control Units (ECUs) and on-board networks. As a consequence, more traditional Electrical/Electronic (E/E) architectures will be replaced by consolidated E/E architectures leveraging heterogeneous computing platforms connected by high bandwidth and low-latency on-board backbone networks [5], [6]. Furthermore, due to complex internal hardware architectures, various levels of shared memories, system buses and inputs/outputs (I/Os), the traditional timing predictability verification techniques are no longer applicable to the software architectures that are deployed on such heterogeneous high-performance platforms, mainly because the worst-case execution times (required by the analysis as one of the inputs) cannot be estimated independent of the response times (an important result provided by the analysis) [7]. What is more, heterogeneous computing platforms open up to the problems of allocation and temporal isolation of software. Additionally, the shift towards more consolidated/clustered architectures leads to the major challenge of supporting and integrating software coming from different application domains, following different models of computation and with different Quality of Service (QoS), real-time and safety requirements [8].

Problem formulation

Considering the importance of timing predictability verification for automotive software systems, there is a need for model-driven methodologies capable of supporting the software development on emerging E/E architectures. However, for this to happen, automotive domain-specific modelling languages must be able to overcome the challenges introduced by emerging E/E architectures; hence, prescribing the type systems and the structure of upcoming heterogeneous software and computing platforms. In addition, either new model-based timing analysis techniques need to be developed or the existing techniques need to be adapted to support the end-to-end timing analysis, which is required to verify the timing predictability of the software architectures on upcoming heterogeneous computing platforms. Furthermore, using the worst-case timing analysis to verify the timing predictability of the software that runs on these advanced computing platforms can render high under-utilisation of the system resources. Hence, an investigation is needed to assess the feasibility and efficacy of the existing worst-case timing analysis when applied to software architectures of the applications that include time-critical as well as performance demanding functionality deployed on heterogeneous computing platforms.

II. Existing MoVES Methodology

Currently, the MoVES methodology is realised by means of six model transformations responsible for translating EAST-ADL models into RCM models where timing analysis can be run, as exemplified in Fig. 1. Within MoVES, EAST-ADL models are used for representing the functional software and hardware architectures, the software timing properties and the software to hardware allocation of automotive software applications. RCM models refine the information described by the EAST-ADL models with timing, control and execution platform information needed for running the model-based timing analysis. The translation step between EAST-ADL and RCM models is characterised by a one-to-many mapping, meaning that one set of EAST-ADL models depicting a given software application could be translated to more than one valid RCM model, representing the same software application. Such a multiplicity is mainly due to the lack of control flow information on the EAST-ADL models. Timing analysis is run on each generated RCM model and the analysis results are fed back to the engineer. However, timing analysis can only be performed on automotive software on distributed E/E architectures leveraging single- or multi-core ECUs. EAST-ADL, RCM, MoVES and all related tools do not support the modelling and end-to-end timing analysis of software architectures for heterogeneous and parallel computing platforms.

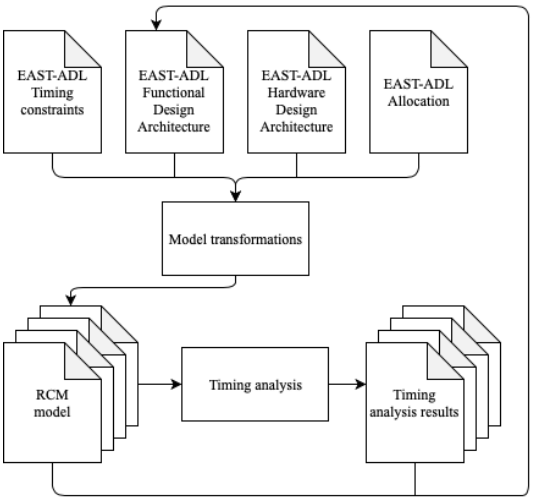

Fig. 1. Simplified representation of the MoVES methodology.

III. Preliminary work

In this ongoing work we discuss a set of improvements to MoVES for supporting automotive software systems on emerging E/E architectures. In particular, we will focus on the following three points in our forthcoming efforts.

Heterogeneous Software: In [9], the authors provided a standard driven software architecture for upcoming automotive software systems realising autonomous driving. One key requirement for modelling languages supporting these architectures will be to address the real-time requirements of applications with different workloads, activation semantics, data flow semantics, real-time characteristics, criticality and safety requirements. We have already extended RCM for supporting different criticality and safety integrity levels (according to the ISO26262 safety standard), activation semantics, and real-time properties and requirements.

High-performance Computing Platforms: Autonomous driving is bringing automotive software systems into the era of high-performance computing. Upcoming automotive platforms will be realised by a combination of automotive-certified real-time micro-controllers and general purpose high-performance computing processors and accelerators [6]. Due to their complex internal hardware architectures, various levels of shared memories and I/Os, heterogeneous platforms can't be verified with traditional model-based timing analysis. What is more, the lack of a reference hardware architecture makes it hard to develop new timing analysis techniques. In this respect, one possible solution would be to provide automotive-specific modelling languages with support for modelling the composing elements of heterogeneous platforms, e.g., GPU, FPGA, etc., and their interaction. In this regard, the language can be equipped with more fine-grained basic blocks such as cores, buses, various types of memories, I/O interfaces, just to name a few, to allow "to model your own heterogeneous computing platform". This approach would also have the benefit of reducing the complexity associated to the hardware modelling by hiding those hardware details which are not primarily concerned with the model-based timing verification.

Model-based Timing Analysis: One important factor that influences the timing analysis of a software architecture is the assumptions about the system model and underlying computing platform. There are different timing analyses for different types of computing platforms. The high levels of heterogeneity and parallelism in the emerging consolidated E/E architectures in the automotive domain makes it more challenging to support the timing analysis of the software architectures. It is likely that adaptation of the existing worst-case timing analysis to support such platforms will result in timing verified software architectures that heavily under-utilise the system resources because the worst-case analyses are based on worst-case assumptions. In this context, we are currently investigating the efficacy of various types of timing analyses.

IV. Conclusion

Recently, several model-driven methodologies have been introduced to manage the software complexity of automotive systems and automate their software development process. MoVES is one such methodology that employs two industrial automotive-specific modelling languages and a set of six model transformations for supporting the development and architectural exploration of these systems with temporal awareness. This paper discusses the challenges and ongoing work with regards to extending the MoVES methodology to support upcoming software architectures for adaptive, autonomous and connected vehicles.

Acknowledgment

The work in this paper is supported by the Swedish Knowledge Foundation (KKS), through the projects A-CPS and HERO, and by the Swedish Governmental Agency for Innovation Systems (VINNOVA), through the projects PANORAMA and DESTINE. The authors would like to thank the industrial partners, especially Arcticus Systems and Volvo, Sweden, for their valuable inputs.

References

[1] A. Bucaioni, L. Addazi, A. Cicchetti, F. Ciccozzi, R. Eramo, S. Mubeen, and M. Sjödin, "Moves: A model-driven methodology for vehicular embedded systems." *IEEE Access*, vol. 6, pp. 6424–6445, 2018.

[2] "East-adl domain model specification, deliverable d4.1.1." (2010), http://www.atesst.org/home/liblocal/docs/ATESST2_D4.1.1_EAST-ADL2-Specification_2010-06-02.pdf.

[3] A. Bucaioni, A. Cicchetti, F. Ciccozzi, S. Mubeen, and M. Sjödin, "Technology-preserving transition from single-core to multi-core in modelling vehicular systems." in *13th European Conference on Modelling Foundations and Applications.*, Springer, Ed., 2017.

[4] The AUTOSAR Consortium, General Specification of Adaptive Platform, Release 19-03, 2019, online: https://www.autosar.org/standards/adaptive-platform/adaptive-platform-1903.

[5] Roland Berger, Consolidation in Vehicle Electronic Architectures, In Think: Aact, Jul., 2015. Available at: https://www.rolandberger.com/en/Publications/pub_consolidation_in_vehicle_electronic_architectures.html, accessed Oct., 2016.

[6] L. Lo Bello, R. Mariani, S. Mubeen, and S. Saponara, "Recent advances and trends in on-board embedded and networked automotive systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1038–1051, 2019.

[7] C. Maiza, H. Rihani, J. M. Rivas, J. Goossens, S. Altmeyer, R. I. Davis, "A survey of timing verification techniques for multi-core real-time systems," Verimag Research Report, Tech. Rep. TR-2018-9, 2018.

[8] S. Saidi, S. Steinhorst, A. Hamann, D. Ziegenbein, and M. Wolf, "Future automotive systems design: research challenges and opportunities: special session," in *Proceedings of the IEEE International Conference on Hardware/Software Codesign and System Synthesis*, 2018, p. 2.

[9] A. C. Serban, E. Poll, and J. Visser, "A standard driven software architecture for fully autonomous vehicles," in *IEEE International Conference on Software Architecture Companion (ICSA-C)*, 2018, pp. 120–127.