Architecting and Analysing Connected Autonomous Vehicles

Alessio Bucaioni*, John Lundbäck§, Patrizio Pelliccione†‡, Saad Mubeen*
* Mälardalen University, Västerås, Sweden; name.surname@mdh.se
† University of L'Aquila, L'Aquila, Italy; patrizio.pelliccione@univaq.it
‡ Chalmers University of Technology | University of Gothenburg, Gothenburg, Sweden
§ Arcticus Systems AB, Järfälla, Sweden; john.lundback@arcticus-systems.com

I. Topic Description

This tutorial focuses on the vehicular domain, which is living a very interesting moment due to the many challenges the domain is experiencing, including autonomy of vehicles, vehicles that are becoming constituent systems in the system-of-systems context and many more. The ever-increasing software complexity in vehicles requires software architecture descriptions, which enable the software developers to compare and relate different products across different vehicle programs, development units, and organisations (in the vehicular ecosystem). Many vehicular functions are constrained by stringent timing requirements. The developers of these functions are required to analyse and verify these requirements at the software architecture level and often very early during the development process [1], [2]. In this context, the tutorial focuses on the design and timing predictability verification of vehicular software architectures for different Electrical and Electronic (E/E) architectures in connected and autonomous vehicles. The key takeaways of the tutorial are: i) an overview of the software development for various vehicular E/E architectures; ii) an overview of state of the art in the area; iii) understanding rudiments and value of timing analysis for this domain; iv) experience an industrial process for architecting and analysing the vehicle software via hands-on practice and demonstration.

II. State of The Art

The works in [3] and [4] propose a common Architecture Framework (cAF), which is used for designing a Domain-specific Architecture Framework (dAF) for vehicular systems. The dAF aims at providing a common ground for documenting, understanding, modelling, analysing, using, and comparing software architectures for a given domain. The dAF for the automotive domain takes the name of Automotive Architecture Framework (AAF). The cAF captures all the concepts that are *common* to any kind of system and organises them into three different views, namely *Functional Architecture*, *Logical Architecture*, and *Technical Architecture*. The functional architecture provides a black-box description of the functional structure of the system in terms of entities, interfaces, interactions among entities, behaviours, and constraints. The work in [5] presents a reference model for the functional architecture for autonomous vehicles, whereas the work in [6] extends the functional architecture for connected vehicles. The logical architecture provides a decomposition of the system into logical components realising the functionalities formalised through the functional architecture. A logical architecture is still hardware agnostic. The dAF in [3] and [4] makes use of only two mandatory views, which are the *Functional Architecture* and the *Technical Architecture*, thus highlighting that in some context the logical architecture might be included in the functional view. The technical architecture specifies how the system functions are implemented from a technical perspective in terms of physical component, geometries and compositions, relationships, dependencies, behaviours, and constraints. In other words, it specifies how the logical components are integrated into a hardware platform and it can be seen as the blueprint for the production of the final product. The technical architecture can have strong correspondences with other optional views. e.g., performance and energy.

EAST-ADL [7] is an architecture description language for vehicular systems. The language and its accompanying development methodology is used by a number of international companies in the vehicular domain [8]. Similar to [3] and [4] EAST-ADL uses a multi-layer approach, where each layer describes the vehicular system at a different abstraction level and from a different perspective. The lower layer in EAST-ADL, namely the implementation layer, is usually delegated to domain-specific languages (DSL), notably AUTOSAR [9], the Rubus Component Model (RCM) [10], and so forth. AUTOSAR was created as an industrial initiative to provide a standardised software architecture at the implementation layer of the EAST-ADL methodology. The timing model for AUTOSAR was developed in the TIMMO and TIMMO-2-USE projects [11]. These projects also developed and later revised the TADL language [12], which is inspired by the MARTE [13] UML profile. The state-of-the-art timing analysis considered in this tutorial corresponds to the end-to-end data-propagation delay analysis of vehicular embedded software systems [1], [14], [15]. The analysis has been implemented in several industrial tools, e.g., Rubus-ICE [16] and SymTA/S.

III. Relevance for ICSA

The main focus of the tutorial is to design vehicular software architectures for various E/E architectures and analyse one of their most important quality attributes, timing. The tutorial will use the principles of model-based software development and component-based software engineering (CBSE) [17] to design the vehicular software architectures. Furthermore, the vehicular domain is of particular interest as it is shifting more and more towards system and software engineering with original equipment manufacturers slowly becoming software companies (nowadays, 90% of innovation in a vehicle comes from software) [6], [18], [19].

The **intended audience** for this tutorial are the academics and practitioners interested in modelling and timing analysis of embedded software architectures. The audience is expected to have an understanding of software architecture, model-driven development, and component-based software development. Basic understanding of real-time systems would be a plus.

IV. Description Of What The Tutorial Will Cover

Introduction. First of all, we will introduce the presenters and discuss the agenda of the tutorial. Thereafter, we will present the learning outcomes and motivation for the tutorial.

Vehicular architectures. This part will discuss three different types of vehicular architectures, namely *distributed*, *domain centralised*, and *vehicle centralised*. These architectures represent the past/present, the present and near future, and the future of automotive architectures. Furthermore, this part will discuss the functional, logical, and technical abstractions of these architectures with the main focus on the technical abstraction.

Timing predictability verification. This part will first introduce the type of timing predictability requirements that can be specified on the vehicular software architectures. Thereafter, we will describe why it is of paramount importance to satisfy these requirements during the software development. This will be followed by a discussion of state-of-the-art techniques and state-of-the-practice engines that are used to analyse and verify these requirements.

Vehicular application use case. This part will present a vehicular application use case from the segment of connected and autonomous vehicles. The purpose of the use case is twofold. First, demonstrate the industrial relevance of the concepts and techniques covered in the tutorial. Second, to provide basis for the hands-on exercise that will be performed by the audience.

Hands-on exercise. The tutorial also includes a hands-on exercise that will be performed by the audience collaboratively in small groups of 3-5 persons. In this exercise, the audience will be asked to analyse and verify timing predictability of a part of the use case (i.e., a simplified software architecture for a distributed E/E vehicular system). The exercise will be manually solved using a pen and paper. The concepts introduced in the previous part of the tutorial and collaborative discussion among the group members will be more than sufficient to be able to solve the exercise. It is likely that different groups may have different (but correct) analysis results based on the assumptions made during the analysis of the software architecture. Such variations in the results due to different practical assumptions will be discussed at the end of this part of the tutorial.

Demonstration and revisiting the exercise. This part of the tutorial will demonstrate the concepts and techniques covered in the tutorial using the Rubus Component Model (RCM) and its accompanying tool chain [20], [21]. RCM and the Rubus tool chain, developed by Arcticus Systems[1], have been used for model- and component-based development of predictable embedded software in the vehicle industry for over 25 years. The vehicular application use case will be modelled and timing analysis using RCM and the tool chain. At this point, the results of the exercise performed by the audience will be revisited and compared with the results computed by the Rubus tool chain.

Conclusion and Discussion. The tutorial will be concluded with a discussion on the tutorial takeaways and addressing final questions from he audience.

[1]https://www.arcticus-systems.com/

Acknowledgement

This work is supported by the Swedish Knowledge Foundation via the project A-CPS, the Swedish Governmental Agency for Innovation Systems (VINNOVA) via the projects PANORAMA & DESTINE, and the Centre of EXcellence on Connected, Geo-Localized and Cybersecure Vehicle (EX-Emerge), funded by Italian Government under CIPE resolution n. 70/2017 (Aug. 7, 2017). We thank the industrial partners Arcticus Systems & Volvo Sweden for their valuable inputs.

References

[1] S. Mubeen, T. Nolte, M. Sjödin, J. Lundbäck, K. Lundbäck, "Supporting timing analysis of vehicular embedded systems through the refinement of timing constraints," *Software & Systems Modeling*, vol. 18, 2019.

[2] 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, 2019.

[3] M. Broy, M. Gleirscher, S. Merenda, D. Wild, P. Kluge, and W. Krenzer, "Toward a holistic and standardized automotive architecture description," *Computer*, vol. 42, no. 12, pp. 98–101, Dec 2009.

[4] M. Broy, M. Gleirscher, P. Kluge, W. Krenzer, S. Merenda, and D. Wild, "Automotive Architecture Framework: Towards a Holistic and Standardised System Architecture Description," Tech. Rep., 2009.

[5] S. Behere and M. Törngren, "A functional reference architecture for autonomous driving," *Inf. Softw. Technol.*, vol. 73, no. C, 2016.

[6] P. Pelliccione, E. Knauss, S. M. Ågren, R. Heldal, C. Bergenhem, A. Vinel, and O. Brunnegård, "Beyond connected cars: a systems of systems perspective," *Science of Computer Programming (SCP)*, 2020.

[7] P. Cuenot, P. Frey, R. Johansson, H. Lönn, Y. Papadopoulos, M.-O. Reiser, A. Sandberg, D. Servat, R. T. Kolagari, M. Törngren, and M. Weber, "The east-adl architecture description language for automotive embedded software," in *Proceedings of MBEERTS'07*, 2010.

[8] 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.

[9] The AUTOSAR Consortium, "Autosar techincal overview, version 4.3." (2016), http://autosar.org.

[10] 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.

[11] "TIMMO-2-USE," https://itea3.org/project/timmo-2-use.html.

[12] "TADL: Timing Augmented Description Language, Version 2, Deliverable 6, October 2009," The TIMMO Consortium.

[13] "The uml profile for marte: Modeling and analysis of real-time and embedded systems." OMG Group, (2010).

[14] N.,Feiertag, K.,Richter, J.,Nordlander, J.,Jonsson, "A Compositional Framework for End-to-End Path Delay Calculation of Automotive Systems under Different Path Semantics." in *Proceedings of the IEEE Real-Time System Symposium, Workshop on Compositional Theory and Technology for Real-Time Embedded Systems,*, (2008).

[15] M. Becker, D. Dasari, S. Mubeen, M. Behnam, and T. Nolte, "End-to-end timing analysis of cause-effect chains in automotive embedded systems," *Journal of Systems Architecture*, vol. 80, pp. 104 – 113, 2017.

[16] 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," in *Computer Science and Information Systems, vol. 10, no. 1, pp 453-482, January 2013*.

[17] T. Vale, I. Crnkovic, E. S. de Almeida, P. A. da Mota Silveira Neto, Y. C. Cavalcanti, and S. R. de Lemos Meira, "Twenty-eight years of component-based software engineering," *Journal of Systems and Software*, vol. 111, pp. 128 – 148, 2016.

[18] S. M. Ågren, E. Knauss, R. Heldal, P. Pelliccione, G. Malmqvist, J. Bodén, "The impact of requirements on systems development speed: a multiple-case study in automotive," *Requirements Engineering*, vol. 24, no. 3, 2019.

[19] P. Pelliccione, E. Knauss, R. Heldal, S. M. Ågren, P. Mallozzi, A. Alminger, D. Borgentun, "Automotive architecture framework: The experience of volvo cars," *Journal of Systems Architecture*, vol. 77, 2017.

[20] A. Bucaioni, A. Cicchetti, F. Ciccozzi, S. Mubeen, and M. Sjödin, "A Metamodel for the Rubus Component Model: Extensions for Timing and Model Transformation from EAST-ADL." *Journal of IEEE Access*, vol. 5, no. 1, 2016.

[21] S. Mubeen, H. Lawson, J. Lundbäck, M. Gålnander, and K. L. Lundbäck, "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 (SER&IP)*, 2017.