Modeling of Vehicular Distributed Embedded Systems: Transition from Single-core to Multi-core

Saad Mubeen
Mälardalen University, Västerås, Sweden
saad.mubeen@mdh.se

Alessio Bucaioni
Arcticus Systems and Mälardalen University, Sweden
alessio.bucaioni@mdh.se

***Abstract*—There are many challenges that are encountered when existing component models, that are originally designed to develop vehicle software for single-core embedded systems, are extended for the software development on multi-core platforms. Within this context, we target the challenge that is concerned with the extension of structural hierarchies in the existing component models. The proposed extensions support the vehicle software development on multi-core platforms, while ensuring backward compatibility with legacy single-core systems, as well as anticipating forward compatibility to future many-core platforms.**

I. Introduction

Majority of functions in modern vehicles are realized by software that runs on Electronic Control Units (ECUs). The size and complexity of the software is continuously increasing due to the high demand for innovations in the vehicle functionality. Already today, the software in a modern car consists of millions of lines of code that runs on tens of distributed ECUs that can be connected by five or more different types of in-vehicle networks [1]. Moreover, many vehicle functions are required to meet real-time requirements, i.e., logically correct functionality should be provided at the times that are appropriate to the function's environment. Such times are dictated by the timing requirements specified on the functions.

Component-based software engineering [2] and model-driven engineering [3], complemented by real-time scheduling theory [4], have proven effective in dealing with the software complexity and real-time challenges in single-core distributed embedded systems in the vehicular domain [1], [5], [6]. Several Component Models (CMs) have been developed in this regard, e.g., AUTOSAR [5], Rubus Component Model (RCM) [6], COMDES [7], just to name a few. However, the existing single-core platforms fall short in providing high computational power to support data-intensive sensors and complex coordination among ECUs that is required to support many advanced vehicle features. Recently, multi-core ECUs have been introduced in the vehicular domain to provide such high levels of computational power [8], [9]. AUTOSAR has recently introduced guidelines to develop multi-core systems [10]. Using these guidelines, the modeling and runtime support for multicore platforms is discussed in [11]. Other frameworks that are in the scope of this work include EAST-ADL [12] and AADL [13]. While the existing software development approaches and tools provide good support for single-core platforms, such a support for multi-core platforms in the vehicle industry is yet to mature.

II. Research Challenges and Paper Contribution

In this paper we identify and target the challenges that are concerned with the extension of the existing CMs (originally designed for signle-core platforms) to support the software development on multi-core platforms. In particular, we target the CMs that explicitly support:

1) separation between the control and data flows among software components,
2) a pipe-and-filter communication style for the interaction among software components.

These CMs allow end-to-end timing analysis of the systems earlier during their development [14], [15]. Besides, these CMs are already used in the vehicle industry. In this work we aim at answering the following question.
"What extensions are needed in the structural hierarchy of the existing CMs to support the vehicle software development on multi-core distributed embedded systems?"

The extensions should ensure backward compatibility with legacy single-core systems, as well as anticipate the vehicle software development on future many-core platforms.

We consider RCM as a starting point for our work. RCM has been used in the vehicle industry for over 20 years, e.g., by Volvo CE[1] and BAE Systems Hägglunds[2]. Currently, RCM supports the development of vehicle functionality only on single-core platforms. We aim to provide a proof of concept for our approach by extending RCM to support the modeling of the software architecture for multi-core platforms as well.

III. Extending the Structural Hierarchy in CMs

The paper aims at achieving several important goals while addressing the question posed in Section II. One goal is to keep the modeling overhead for the user as small as possible. In other words, the extended CM for multi-core platforms should not enforce a lot of extra work for the user who already uses the existing CM for developing the software on single-core platforms. Another goal is to support the software development on legacy single-core and contemporary multi-core ECUs, while anticipating further extensions of the CM to support future many-core ECUs that will contain several tens of cores connected by on-chip networks.

A. Structural Hierarchy in CMs supporting Single-Core ECUs

The structural hierarchy, shown in Fig. 1, is found in the majority of CMs that use the pipe-and-filter communication and distinguish between the control and data flows among software components in the software architecture of single-core systems [6], [7], [16]. In this structure, the highest-level hierarchical element is called the $system$. The $system$ contains the models of one or more networks and nodes (ECUs). A network contains the models of Network Specification (NS) and message objects. The NS is the model representation of a

[1]https://www.volvoce.com
[2]http://www.baesystems.com

physical network. It is unique for each network communication protocol, e.g., CAN, Flexray and switched Ethernet. Each message contains a set of signals that are mapped to it.

The node contains one or more models of processor. Each processor defines a runtime environment for the node. Basically, a processor represents the hardware and operating system specific instance of a node. Several processors can be assigned to a single node, e.g., a real hardware target such as the ARM processor or a virtual processor that simulates the instruction set and environment of the hardware target. Note that only one processor can be selected from each node during the deployment. The node also includes an interface that contains one or more network ports that are responsible for sending/receiving messages to/from the network respectively. Each processor contains one or more models of modes. A mode defines different states of the system. A mode may contain one or more composites. Each composite is a container that encapsulates one or more Software Components (SWCs). An SWC is the lowest-level hierarchical element that encapsulates basic functions and has run-to-completion semantics. Each SWC contains only one interface and one or more behaviors. The interface contains only one input trigger port and one or more output trigger ports, input data ports and output data ports. The behavior represents a function, e.g., a C function or a simulink block.

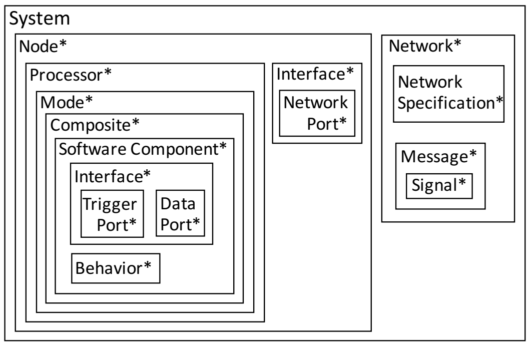

Fig. 1. Structural hierarchy in the existing component models for single-core vehicular distributed embedded systems.

B. Proposed Extensions to the Structural Hierarchy

In order to support seamless software development of the systems on single-core as well as on multi-core platforms, we propose extensions to the structural hierarchy of the existing CMs by introducing the models of core, partition and Intra-Processor Communicator (IPC) as shown in Fig. 2. Note that the parts of the structure that are above the processor and below the mode in the extended hierarchy remain the same. The model of a processor in the extended hierarchy contains one or more cores. A core contains at least one partition. Each partition is assumed to run an instance of the operating system. Each processor also contains one model of the IPC object which handles inter-core communication as well as inter-partition communication within each core. The IPC object can be adapted for any inter-core communication platform, e.g, cores communicate via direct connections, cores communicate via a bus (in the case of multi-core platforms), and cores communicate via a network (in the case of many-core platforms). The structure in Fig. 2 is sufficient to support the software development for multi-core ECUs. Moreover, the structure can be used to develop the vehicle software on single-core ECUs by setting the number of cores and partitions each equal to one. The structure also allows to reuse the complete models of single-core legacy nodes by allocating the legacy processor to the partition in a single-partition core. The proposed partition model is inline with the partition concept in the ARINC 653 Avionics standard [17]. Hence, the partitions support the development of single- and multi-core systems that have different criticality levels in their software architectures.

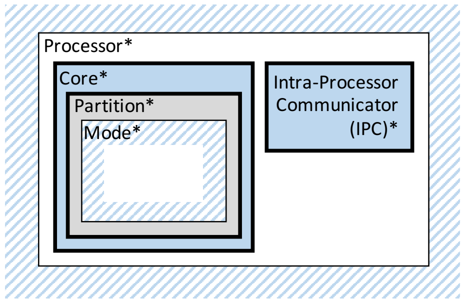

Fig. 2. Proposed extensions to the structural hierarchy of existing component models to support multi-core vehicular distributed embedded systems.

C. Summary of Ongoing Work

Currently, we are developing a hardware model corresponding to the software hierarchy in Fig. 2. We also plan to provide a software-to-hardware allocation model. Further, we will provide a proof of concept for the proposed extensions.

Acknowledgement

The work in this paper is supported by the Swedish Knowledge Foundation within the project PreView. We thank all our industrial partners, especially Arcticus Systems, Volvo CE and BAE Systems Hägglunds, Sweden.

References

[1] M. Broy, I. Kruger, A. Pretschner, and C. Salzmann, "Engineering automotive software," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 356 –373, Feb. 2007.
[2] I. Crnkovic and M. Larsson, *Building Reliable Component-Based Software Systems*. Norwood, MA, USA: Artech House, Inc., 2002.
[3] D. C. Schmidt, "Guest editor's introduction: Model-driven engineering," *Computer*, vol. 39, no. 2, pp. 25–31, Feb. 2006.
[4] N. Audsley, A. Burns, R. Davis, K. Tindell, and A. Wellings, "Fixed priority pre-emptive scheduling: an historic perspective," *Real-Time Systems*, vol. 8, no. 2/3, pp. 173–198, 1995.
[5] "AUTOSAR Techincal Overview, Release 4.1, Rev. 2, Ver. 1.1.0., The AUTOSAR Consortium, Oct., 2013," http://autosar.org.
[6] K. Hänninen et.al., "The Rubus Component Model for Resource Constrained Real-Time Systems," in *3rd IEEE International Symposium on Industrial Embedded Systems*, Jun. 2008.
[7] X. Ke, K. Sierszecki, and C. Angelov, "COMDES-II: A Component-Based Framework for Generative Development of Distributed Real-Time Control Systems," in *13th International Conference on Embedded and Real-Time Computing Systems and Applications*, Aug. 2007.
[8] "SymTA/S for Migration from Single-core to Multi-core ECU-Software on Infineon Microcontrollers, Electronic Engineering Journal, Dec., 2010."
[9] 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.
[10] "Guide to Multi-Core Systems, Release 4.1, Rev. 3, The AUTOSAR Consortium, Mar., 2014," https://www.autosar.org.
[11] A. S. Moghaddam, "Performance Evaluation and Modeling of a Multicore AUTOSAR System," Master's thesis, Department of Computer Science and Engineering, Chalmers University of Technology, Sweden, May 2013.
[12] "EAST-ADL Domain Model Spec., V2.1.12, acessed Nov., 2016," http://www.east-adl.info/Specification/V2.1.12/EAST-ADL-Specification_V2.1.12.pdf.
[13] 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*. Springer US, 2005, vol. 176, pp. 3–15.
[14] 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.
[15] 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.
[16] S. Sentilles, A. Vulgarakis, T. Bures, J. Carlson, and I. Crnkovic, "A Component Model for Control-Intensive Distributed Embedded Systems," in *the 11th International Symposium on Component Based Software*, 2008, pp. 310–317.
[17] "ARINC Specification 653P1-2, Avionics Application Software Standard Interface Part 1: Required Services." http://www.arinc.com.