HW Componentizing Kernel: A New Approach to address the Mega Complexity of Future Automotive CPS

Jong-Chan Kim, Kyoung-Soo We, and Chang-Gun Lee *

1 Introduction

Automobile is an important application of CPS (Cyber Physical System). However, current software development process in the automotive industry is not adequate to solve the unique problems of CPS. This paper pinpoints the limitations of the current automotive software development process in the perspective of CPS and proposes a new kernel-based approach called *HW componentizing kernel* as a solution.

2 Background, Motivations, and Challenges

Due to the distributed nature of automotive systems, automakers are forced to integrate software components independently-developed by different suppliers. Therefore, the design phase, in which overall architectures and requirements are fixed and verified, has become the most important part of overall system development process.

In the design phase, automakers define the architecture and requirements, especially for the timings, as follows: First, the overall system design is described as a chain of software components. Then, a delay requirement bound as well as functional requirements is defined for each software component. To ensure that the design safely meets the system's timing requirements, the sum of delay bound requirements of software components in the component chain is kept smaller than the end-to-end delay requirement of the component chain.

Then, tier-1 suppliers develop softwares conforming to the functional requirements and the delay bound requirement. After the development and the unit test phase, outputs are delivered to automakers in the form of ECUs, not in the form of independent softwares. Finally, automakers actually integrate ECUs. During the integration phase, the execution time of each software component does not change since each software component is physically isolated and does not interfere with each other. Therefore the final system behaves exactly the same as intended by the system designer.

*J.-C. Kim, K.-S. We, and C.-G. Lee ({jongchank, we123456, cglee}@snu.ac.kr) are with the School of Computer Science and Engineering, Seoul National University, Seoul. The corresponding author is C.-G. Lee. The authors can be contacted by phone at 949-302-8105.

Current design process stated above is well suited to today's vehicle architectures where each ECU is designed to serve only a single function. However, automakers now seek for a new design and integration method to provide more advanced features in less ECUs for at least four reasons: (1) Manufacturing cost is an important competitiveness factor; (2) To provide more space to passengers; (3) Wire harness which connects ECUs, sensors, and actuators is becoming too complex with increasing number of ECUs; and (4) Safety features are combining to form advanced safety systems rather than working separately. Consequently, each ECU should perform multiple dependent/independent functions concurrently in the future vehicles.

Unfortunately, however, today's RTOS does not guarantee that each software component in the execution environment behaves exactly the same as intended by the system designer because of the delays caused by higher-priority components. This makes it very difficult to adopt RTOS in safety-critical parts such as chassis controls and safety systems. Therefore, a new design method and an execution architecture other than RTOS are required.

3 HW Componentizing Kernel Approach

In this section, we propose a HW componentizing kernel approach as a solution to the challenges described in Section 2. We start with the design principles of our approach which follows: (1) The kernel transparently translates the high-level component-based designs into execution environments; (2) The integration phase does not involve additional mappings between logical design entities (i.e., software components) and physical execution entities such as tasks and threads; (3) The functional and non-functional properties of each software component do not change during the integration phase; and (4) The integrated system behaves exactly the same as the system designer's intention.

To satisfy the design principles of the HW componentizing kernel stated above, the high-level component-based design specifications are passed to the kernel through a newly defined set of APIs called *componentizing API*. Then, the *componentizing scheduler* actually partitions and componentizes hardware resources such as CPU and RAM into smaller ones called *hardware components* that provide guaranteed performance or capacity to fulfill the delay bound requirements of the software components. As a result, the HW componentizing kernel guarantees that each software component bundled with a hardware component behaves as the system designer's intention ensuring the integrated system's timing requirements. The rest of this section briefly introduces the componentizing API and the componentizing scheduler.

Componentizing API: Our kernel is aware of the high-level component-based design specifications through the componentizing API. Componentizing API gathers information including: (1) Per-component properties (i.e., execution time, delay bound requirements, program codes to execute, and the hardware resource to be run on); (2) Inter-component interfaces which connect components' input ports and output ports; and (3) Component relation graphs which describe the precedence relations among software components. Then, the program codes are loaded to create each component. The input ports and output ports of each component are connected to form the component chain.

Componentizing Scheduler: Componentizing scheduler, which is aware of each component's timing requirements through the componentizing API, provides a dedicated hardware component with an appropriate proportional performance or capacity to each software component. A proportional-share algorithm actually partitions a hardware resource into multiple hardware components. As a result, each software component feels that it is executing on a dedicated hardware resource (i.e., hardware component) which is slower than the real hardware resource but able to meet the timing requirement of the software component.

4 Conclusion

This paper presents challenges of future automotive software development in the perspective of CPS and proposes a HW componentizing kernel approach as a solution. In our HW componentizing kernel approach, the high-level component-based design is transparently translated into execution environments by dedicating componentized hardware resources to each software component. Thus, non-functional properties such as timings of software components as well as functional properties are preserved during their integration.

Our componentizing kernel approach eventually provides physical *compositionality* [1] and *composability* [2] of functional and non-functional properties of software components. To overcome the mega complexity of future automotive CPS, a physical composition technology as well as a logical composition methodology is essential.

Biographies

Jong-Chan Kim is a Ph.D. candidate in the School of Computer Science and Engineering at Seoul National University, Korea and is currently working as a visiting researcher at Hyundai Mobis, Korea. His current research is focused on the software development process of future automotive CPS. **Kyoung-Soo We** is a Ph.D. student in the School of Computer Science and Engineering at Seoul National University, Korea and is currently working as a visiting researcher at Hyundai Mobis, Korea. He is currently working on a new software integration method for automotive softwares. **Chang-Gun Lee** is currently an Associate Professor in the School of Computer Science and Engineering, Seoul National University, Korea and also a Visiting Professor at University of California, Irvine. Previously, he was an Assistant Professor in the Department of Electrical and Computer Engineering, The Ohio State University, Columbus from 2002 to 2006, a Research Scientist in the Department of Computer Science, University of Illinois at Urbana-Champaign from 2000 to 2002, and a Research Engineer in the Advanced Telecomm. Research Lab., LG Information and Communications, Ltd. from 1998 to 2000. His current research interests include real-time embedded systems, cyber-physical systems, ubiquitous systems, QoS management, wireless ad-hoc networks, and flash memory systems.

[1] System-level properties can be computed from local properties of components.

[2] Component properties are not changing as a result of interactions with other components.