Model-Driven Development for Safety-Critical Software Components

By Franz Walkembach, Product Line Manager
EXECUTIVE SUMMARY

Software platforms are becoming an increasingly important part of electronics development in the automotive industry. Software-driven functionality—such as advanced driver assistance systems (ADAS), external connections to the Internet of Things (IoT), and autonomous driving—is increasingly being integrated in today’s and future cars, which can have a major impact on safety and certification. On average, a typical car today has 30 dedicated electronic control units (ECUs), and some higher-end vehicles have as many as 120 or more. These ECUs entail a significant bill-of-materials and, with all the associated cabling and housing, take up an increasing amount of space, in addition to drawing power and adding weight that will decrease the energy efficiency of the vehicle.

Historically, the automotive industry has taken a federated approach, with each car function implemented in a separate box; but the growing trend now is to integrate more and more functions onto fewer ECUs. But this multitude of ECUs will likely come from many different suppliers, posing a significant development issue for automotive original equipment manufacturers (OEMs): how do these ECUs communicate with each other, and how can the system development be managed effectively?

Automotive Open System Architecture (AUTOSAR) is a major industry initiative to manage the growing electrical and electronic complexity in vehicles through the increased reuse and transferability of software modules. This paper provides an overview of AUTOSAR and its model-driven development approach, discussing its limitations and how timing behavior is a critical element for increased design flexibility in next-generation automotive systems.

TABLE OF CONTENTS

Executive Summary ................................................................. 2
AUTOSAR ................................................................. 3
Timing Behavior ................................................................. 3
Transformation ................................................................. 4
Automotive Profile for VxWorks .............................................. 5
Conclusion ................................................................. 5
AUTOSAR

Set up in 2003 as a worldwide development partnership between vehicle manufacturers, automotive OEMs, and other companies from the electronics, semiconductor, and software industries, AUTOSAR has been working on the development of open and standardized software architectures for ECUs.

The standard comprises specifications that describe software architecture, application interfaces, and a development methodology. The layered software architecture (see Figure 1) enables the development of abstracted software components, known as “SwComponentTypes” in AUTOSAR terminology. These software components model car functions and are implemented independently from underlying hardware. They can therefore be used across ECUs from different vendors, and can also span multiple ECU product generations. The virtual function bus (VFB) is used to assemble and integrate these components into a virtual system and to validate communication consistency between components. Transformation

Later in the development process, the model is transformed to a network of ECUs, which are interconnected via the CAN, LIN, or FlexRay bus, for example. AUTOSAR also defines the methodology and tools required to bring information from the various elements together, including ECU and system-constraint descriptions, to perform this transformation and map software components to a system of ECUs. It also includes the mapping configuration of the “basic software”—running below the AUTOSAR run-time environment—on each of the ECUs. Within the system, there are different domains (such as the powertrain for engine control or body control for window/mirror functionality, for example), which are connected via network gateways. In essence, it means that a software application developer does not need to be aware of which ECU the application is running on.

Key advantages to the AUTOSAR approach are that all manufacturers and developers have a common idea about what an automotive electrical or electronic system should look like, and that independent software modules and elements are highly modular, scalable, transferable, and reusable. This AUTOSAR model is essentially an abstraction of functionality specifications, and can also be used for simulation in the early stages of development to test software and ensure expected behavior. However, it may not be a complete model of the system, and if it is incomplete, then the results of simulation will be likewise.

TRANSFORMATION

Later in the development process, the model is transformed to a network of ECUs, which are interconnected via the CAN, LIN, or FlexRay bus, for example. AUTOSAR also defines the methodology and tools required to bring information from the various elements together, including ECU and system-constraint descriptions, to perform this transformation and map software components to a system of ECUs. It also includes the mapping configuration of the “basic software”—running below the AUTOSAR run-time environment—on each of the ECUs. Within the system, there are different domains (such as the powertrain for engine control or body control for window/mirror functionality, for example), which are connected via network gateways. In essence, it means that a software application developer does not need to be aware of which ECU the application is running on.

Key advantages to the AUTOSAR approach are that all manufacturers and developers have a common idea about what an automotive electrical or electronic system should look like, and that independent software modules and elements are highly modular, scalable, transferable, and reusable. This AUTOSAR model is essentially an abstraction of functionality specifications, and can also be used for simulation in the early stages of development to test software and ensure expected behavior. However, it may not be a complete model of the system, and if it is incomplete, then the results of simulation will be likewise.
TIMING BEHAVIOR

In model-driven development like AUTOSAR, the aspects of the model and those of the ECU implementation must be strictly separated, especially if there is a future intention to apply the model to a different implementation. With respect to formally guaranteeing correctness of timing behavior, the AUTOSAR model is incomplete. In C code development there is no language construct that enables the modeling of timing behavior. This can impact communication between ECUs: for example, access to automotive environment sensors needs to be provided in a scalable, timely, and predictable way. Changes in functionality during development will influence timing behavior, and the complete system will need to be validated repeatedly, significantly increasing system architecture integration effort. In essence, there is no predictable software behavior, and AUTOSAR-compliant tools will need to be used to perform scheduling analysis and timing verification.

A different approach, adopted by Wind River®, is to make the design more predictable with regard to timing behavior. Using an AUTOSAR-compliant toolchain for modeling software components, timing information can be augmented to the software components with the addition of a Timing Definition Language (TDL) construct to the C code design. Due to this time and value determinism, the same input values imply correspondingly the same output values with respect to time, which significantly improves reliability. The TDL is based on the concept of the logical execution time (LET) (see the example shown in Figure 2), which abstracts the physical execution time on a particular platform and the communication topology, and allows the simulation of applications while maintaining real-time behavior.

It means that components can be developed independently, providing real abstraction from both the hardware and software platform, so the impacts of extended functionality can be seen and dealt with immediately.

The addition of the timing description enables the simulation and testing of the model and the ”attaching or scattering” of the software components to the different physical targets without impacting the timing behavior. This provides the system developer with predictable software integration with the same behavior on the ECU deployment as seen in the simulation stage.

---

**Figure 2. Logical execution time**

The addition of the timing description enables the simulation and testing of the model and the ”attaching or scattering” of the software components to the different physical targets without impacting the timing behavior. This provides the system developer with predictable software integration with the same behavior on the ECU deployment as seen in the simulation stage.
Enabling the distribution of components across different network nodes without affecting overall system behavior, OEMs can move to the system architecture stage, either via the federated approach with scattering on multiple ECUs, or taking a more consolidated approach to fewer and more highly integrated systems (see Figure 3).

Overall, the main advantage of this approach is that timing behavior is the same in both the simulation and deployment stages, as well as there being no difference in a local or distributed execution of software components on single or multi-core ECU designs.

AUTOMOTIVE PROFILE FOR VXWORKS

Engineered in conformance with AUTOSAR standards and methodology, Automotive Profile for VxWorks®, the Wind River industry-leading real-time operating system (RTOS), provides an ISO 26262-certifiable platform and AUTOSAR-compatible run-time environment that supports standardized connectivity and functional interfaces to other automotive software components, enabling simpler, cheaper, and faster interoperability and integration. Automotive Profile enables the reliable consolidation of a large number of software-driven functions on a smaller number of more powerful ECUs.

The Wind River development methodology (see Figure 4) can enable the smooth transition of legacy systems to multi-core ECU systems. Using space/time partitioning, and leveraging LET, OEMs can consolidate multiple vehicle controls with different levels of safety criticality up to ASIL D onto a single hardware platform while maintaining time- and space-based separation and isolation. This ability introduces the potential for reductions in cost and weight in vehicles, and can help mitigate the risk of attack or interference to other software components without compromising vehicle safety or functional performance. It addresses the growing needs for safe, secure, and certifiable software-driven ADAS and autonomous driving applications.

CONCLUSION

As the number ECUs in the car increase, OEMs need to find a way to consolidate and organize their development process to manage the complexity. In an AUTOSAR-based architecture with abstraction of functionality, often the formally guaranteeing correctness of timing is missing when it comes to system updates. This becomes critical in a certified environment or to address safety-critical components for ADAS or autonomous driving. One way to address this is via a TDL and a model-driven development environment.

Automotive Profile includes a design methodology to combine the VxWorks RTOS with AUTOSAR including TDL. It provides an ISO 26262-certifiable platform and exposes VxWorks system calls for a model-based development that supports standardized connectivity and functional interfaces to other automotive software components. This way space/time partitioning and logical execution time can be combined and used as a development environment for a high number of ECUs with different system suppliers delivering the ECU systems.