Automotive electronics and software development
White paper
July 2009

Rational. software

IBM

Modeling AUTOSAR systems with a UML/SysML profile.

Developing automotive systems and software using UML/SysML

Richard F. Boldt, IBM Software Group, Rational
Senior Product Manager, Systems Modeling Products

Contents

The AUTOSAR initiative brings together leading automotive manufacturers and suppliers to define standards that enable reuse and scalability of software and hardware across different platforms.

Introduction

Developing today's automobiles requires a seemingly ever-increasing number of software and electronics applications for each automotive platform. Owing to the increased cost of the electronics and the ability of the electronic features to differentiate one vehicle from another, the automotive industry is scrutinizing how the electronics are developed and deployed within an automobile. In response to this challenge, the AUTOSAR initiative was created.

The AUTOSAR initiative has brought together leading automotive manufactures and suppliers in a collaborative effort to define standards that will allow the reuse of hardware, software and architecture from one vehicle to another, while allowing the individual companies to maintain their distinctive intellectual property.

The goals of the AUTOSAR initiative are to create standard core solutions that enable the reuse and scalability of functional modules between platform variants. This includes different vehicles, together with whole product lifecycle support, plus software updates and upgrades over the vehicle lifetime while also enabling an increased use of commercial off the shelf hardware (COTS). The AUTOSAR initiative covers all vehicle domains and electronic applications; consequently, the standard must address critical industry demands, such as reducing costs, allowing for the transferability of functions throughout the network, integrations from multiple suppliers and safety considerations.

To address these needs, the AUTOSAR initiative has developed a layered software development approach that allows companies to cooperate on standards and compete on implementations. In order to implement an AUTOSAR process, however, it is critical to define the AUTOSAR system. Although this can be done in several ways, a powerful and extensible way is to use a Unified Modeling Language (UML)/Object Management Group (OMG) Systems Modeling Language (SysML) profile tailored to meeting the AUTOSAR standards.

Highlights

This paper provides an overview of the fundamental concepts of AUTOSAR such as the AUTOSAR System, AUTOSAR Software Components (SW-Cs), the AUTOSAR Virtual Functional Bus (VFB) and the AUTOSAR Run Time Environments (RTE). An AUTOSAR System consists of a set of interacting AUTOSAR Software Components communicating via the AUTOSAR VFB. This layered approach allows the AUTOSAR SW-Cs to be mapped to a specific Electronic Control Units (ECU) in a way such that these elements can be distributed over an ECU network, while remaining correctly integrated to enable the reuse of software assets from one platform to another. The AUTOSAR VFB defines a conceptual communications framework from one AUTOSAR SW-C to another, while the AUTOSAR RTE is a concrete implementation of the AUTOSAR VFB.

In addition to covering the basic concepts of AUTOSAR, this paper reviews the details of a UML/ SysML based AUTOSAR profile for capturing the AUTOSAR System—including the interacting AUTOSAR Software Components—and discusses how using UML the AUTOSAR System Model can be extended to include effective model organization, requirements capture and behavioral/ algorithm modeling.

The AUTOSAR process is mapped out to include the entire software specification procedure.

Mapping the AUTOSAR process

The AUTOSAR process is mapped out to include the entire software specification procedure, from the highest levels of abstraction to the details of scheduling different code fragments of an individual software component on an ECU. For example, at the top level, there is the functional system that includes all the features that make up the different electrical sub-systems such as a wiper system, transmission control system or the exterior lighting system of the automobile.

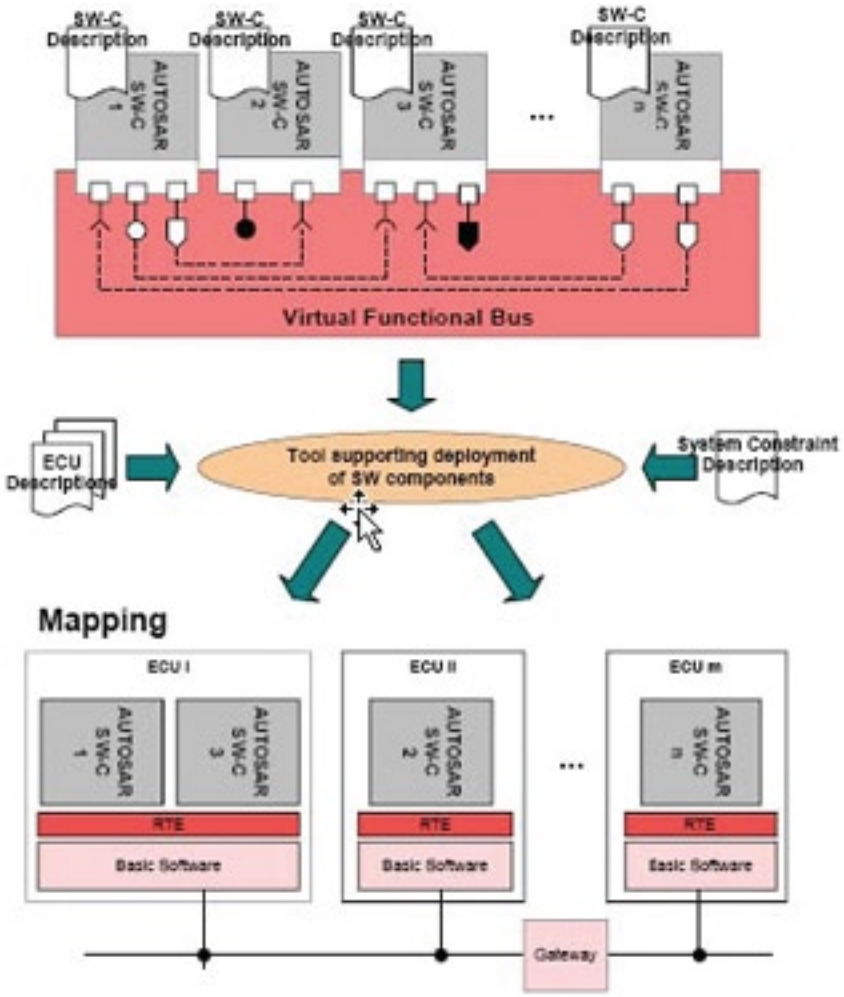

Figure 1: Basic AUTOSAR approach

These systems consist of elements called AUTOSAR Software Components; at the lowest level, the AUTOSAR SW-C is an atomic unit that completely encapsulates its data, has well defined interfaces and communicates with the AUTOSAR VFB/RTE and must be implemented in a single ECU. At its lowest level, an atomic AUTOSAR SW-C cannot be spread across multiple ECUs.

The AUTOSAR Virtual Functional Bus (VFB) abstracts the communication between AUTOSAR SW-Cs away from the hardware in a technology neutral way, so that it does not matter if the communication between two software components will occur within the same ECU or over a bus between multiple ECUs. The VFB allows for two ways of communication between AUTOSAR SW-Cs: sender-receiver mode, where one component broadcasts out information and other components are listening to receive it, or a client-server mode, where the client asks for something and the server provides the requested service.

Defining the topology

The next step in the process is to define the electrical architecture (topology) for a specific vehicle. The electrical architecture consists of the ECUs, the busses (CAN, LIN, FlexRay and MOST) and gateways that will connect the ECUs together. Once the functional systems and the electrical architecture are designed, the individual AUTOSAR SW-Cs captured in the functional system can be mapped to the ECUs they reside in within the electrical architecture. With mapping now complete, a communication matrix characterizing the information that is passed over the different busses can be derived.

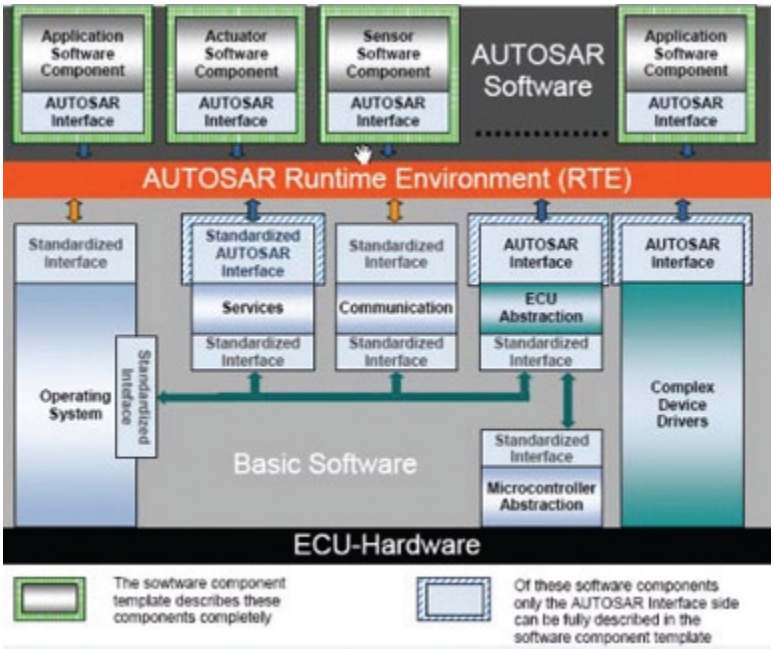

Figure 2: ECU software architecture

Highlights

The diagram above shows the AUTOSAR layered software system within a single ECU. The software is broken up into three primary layers. The application software that consists of the different AUTOSAR SW-Cs that have been allocated to this specific ECU, the AUTOSAR Basic Software, which provides the interaction with the hardware and the AUTOSAR Runtime Environment (RTE) that sits between the application software and the basic software.

By standardizing the interface between the software components and the basic software, the software components can easily be retargeted to different ECUs within dissimilar electrical architectures.

The RTE, a realization of the VFB, is a set of APIs and middleware that connects the Basic Software to the SW-Cs. Everything contained below the RTE is called Basic Software, whose purpose is to abstract away the hardware and includes the operating system (perhaps OSEK), the communication interfaces (perhaps CAN, LIN, FlexRay or MOST), the device drivers and other services. The net effect is that by standardizing the interface between the software components and the basic software, the software components can be retargeted easily to different ECUs within dissimilar electrical architectures.

A powerful means to capture an AUTOSAR system is by using graphical models. This provides the benefits of abstraction from a design standpoint and makes it easier to communicate the design intent. When selecting a graphical modeling notation, basing it on a widely used standard makes great practical sense. If this standard also contains elements that map well to the elements defined in AUTOSAR, and they allow for a mechanism to be tailored to meet domain-specific needs, then the combined benefits of a robust language can be achieved.

Within the context of AUTOSAR, the combination of UML and SysML modeling provide the benefits of abstraction from a design standpoint while making it easier to communicate the design intent.

UML/SysML-based AUTOSAR

Within the context of AUTOSAR, the combination of UML and SysML meets all these criteria. A UML/SysML-based AUTOSAR profile using domain-specific terminology to express the software components, interfaces, ECUs and electrical architectures for automobiles can be created using five diagrams, each of which is outlined below.

- *Software Component Diagram is used to define the functional software architecture by defining the different AUTOSAR SW-Cs and how they communicate with each other.*

- *Internal Behavior Diagram is used to specify the scheduling of various runnable entities within a specific AUTOSAR SW-C and how this will integrate with the AUTOSAR RTE.*
- *ECU Diagram is used to define the ECU types and their communication ports.*
- *Topology Diagram is used to define the physical topology or electrical architecture of the system including all the ECUs in the automobile and how they are connected.*
- *Systems Diagram is used to capture the overall AUTOSAR system including how the AUTOSAR SW-Cs map to the individual ECUs that make up the electrical architecture and the communication matrices between the ECUS.*

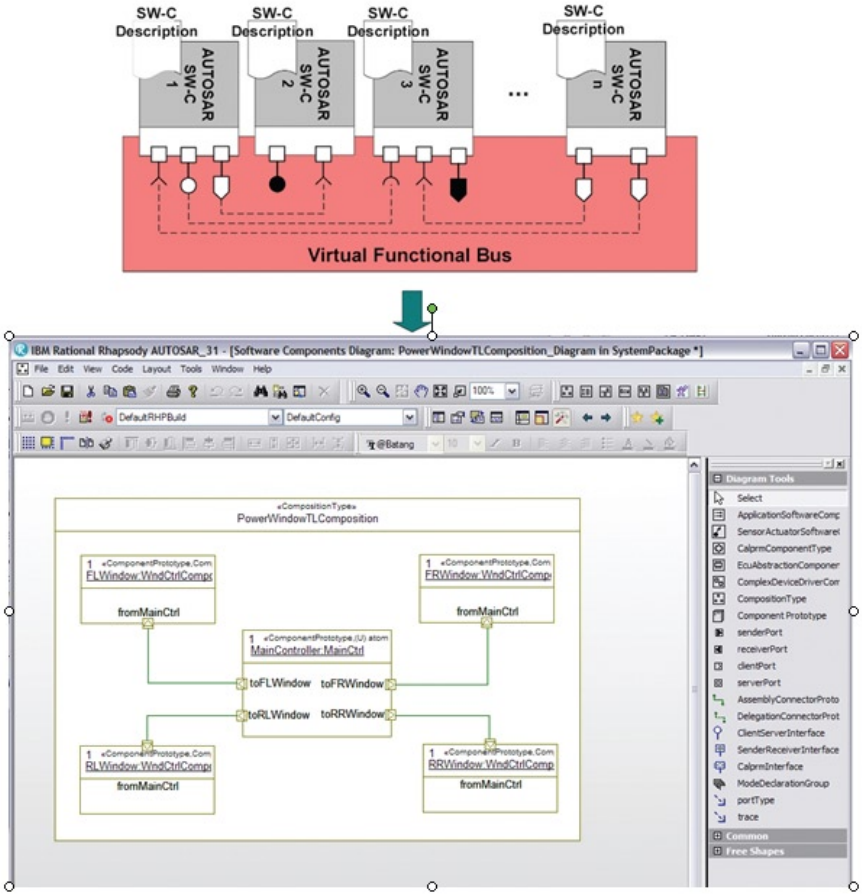

Figure 3: Software component diagram

Defining the diagrams

The software component diagram's foundation is based on a combination of the UML/SysML class and object diagrams. This diagram enables engineers to capture individual software architectures of the functional system by defining the individual software components and the communication between the components over the VFB.

The functional system consists of the different electrical sub-systems in the car such as security, stability control or exterior lighting that are represented as a software composition and the AUTOSAR SW-Cs that make up each of these sub-systems such as fog lights, high beams, turn signals, hazards, front right light unit control and left rear light unit control. The communication between the different AUTOSAR SW-Cs is defined using server ports, client ports, sender ports and receiver ports. These are all specializations of the UML concept of ports. Sender ports produce data that is consumed by receiver ports and client ports request services provided by server ports.

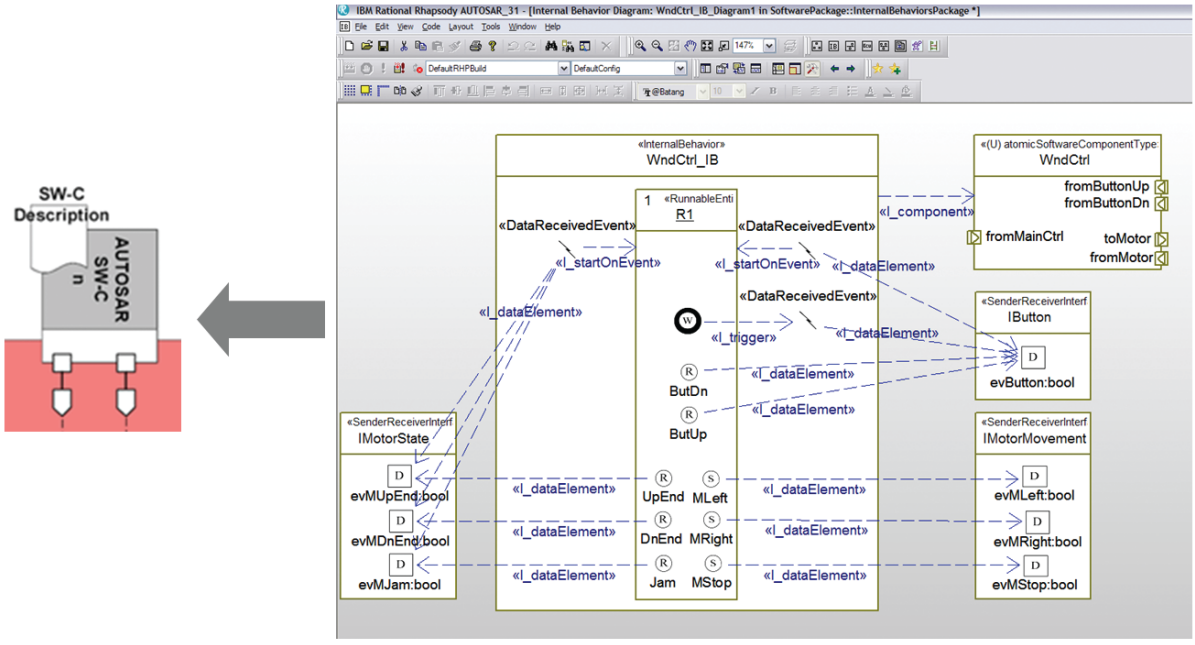

Figure 4: Internal behavior diagram

Highlights

Software components are made up of various runnable entities that can be executed and scheduled independently from the other runnable entities.

The internal behavior diagram specifies the internal scheduling of runnable entities and the interface to the RTE for a specific software component, and is based on a combination of a class and object diagram.

A software component is made up of various runnable entities. A runnable entity is a code fragment, algorithm (logical or computational), function or combination of one or more of these or similar items that can be executed and scheduled independently from the other runnable entities. Thus, a runnable entity needs to be scheduled to execute.

A scheduling event is defined as an event that is caused by a trigger, such as a signal occurring, a timer expiring or a data item crossing a threshold. The different runnable entities that make up an AUTOSAR SW-C and their scheduling is defined in the internal behavior diagram but the actual implementations (code bodies or the algorithms themselves) are not expressed in this diagram. These are either directly captured in code or defined using a behavioral modeling tool (BMT).

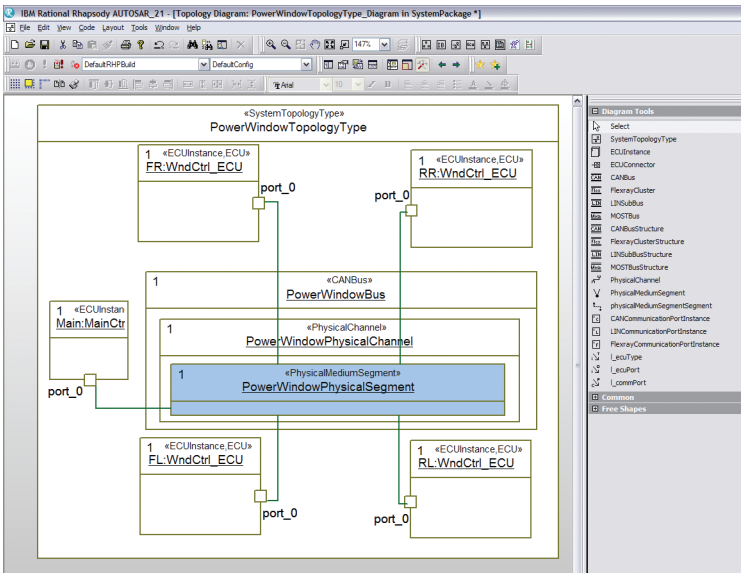

Figure 5: Topology diagram

Highlights

The ECU diagram describes the hardware configuration details for each individual ECU type and its ports. The topology diagram is used to define the physical (electrical) architecture of the system.

The ECU diagram captures the ECUs by describing the hardware configuration details for each individual ECU type and its ports. The topology diagram is used to define the physical (electrical) architecture of the system—for example the ECU instances (described in ECU diagrams) and the busses that connect them. The ECU diagram and the topology diagram are based on a combination of the UML/SysML class and object diagrams and allow for the definition of the various ECUs, their ports and the CAN, LIN, FlexRay and/or most busses that connect them.

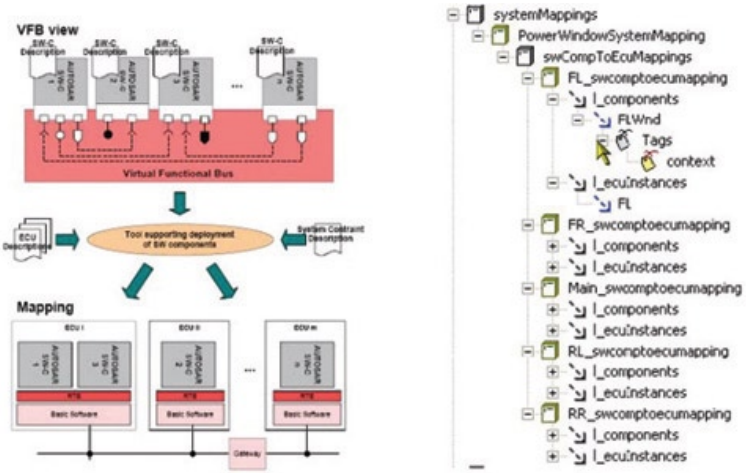

Figure 6: Systems diagram contains the mapping of SW-C to ECUs

The system diagram is used to define a complete AUTOSAR system. This is achieved by defining the mapping of the AUTOSAR SW-Cs defined in the software component diagram to the electrical architecture as defined in the topology diagram for a specific vehicle by defining the ECU that each AUTOSAR SW-C will be implemented in. Once this mapping is preformed the bus communication matrix for the vehicle can be derived.

UML and SysML can be used to go beyond what is covered in AUTOSAR, especially in requirements capture and elaboration, behavioral modeling and design organization.The

An additional benefit of using UML and SysML as the underlying mechanism to define the AUTOSAR modeling environment is that UML and SysML can be used to go beyond what is covered in AUTOSAR, especially in the areas of requirements capture and elaboration, behavioral modeling and design organization. The UML/SysML concept of packages can be used to group model elements together in different ways, while components can be used to define various deployable configurations, providing model flexibility and organization.

Highlights

One area that AUTOSAR does not cover in detail is the process of capturing and elaborating on the system requirements. Here, UML and SysML can play an especially important role. The SysML requirements diagrams can be used to capture the textural requirements, and then leveraged to trace those requirements to the AUTOSAR SW-Cs, runnable entities and the ECUs that will implement these requirements. Additionally, the requirement can be linked to the test scenarios that are used to validate the system. Using UML/ SysML Use Case diagrams enables the requirements organization to describe the different system uses and define them even further, based on the typical use cases for the system, while sequence diagrams can be used to describe the different scenarios that the system is required to perform. Finally, UML/ SysML Statecharts and Activity diagrams provide powerful means to express the behavior of the different runnable entities that make an AUTOSAR SW-C.

Conclusion

Today's automobiles are dependent on software and electronics applications for their reliability, functionality, and even marketability on the showroom floor. Due to cost pressures and the ability of the electronic features to differentiate one vehicle from another, it makes sense that the industry is scrutinizing how the electronics are developed and deployed within an automobile—ultimately resulting in the AUTOSAR initiative. The AUTOSAR initiative covers all vehicle domains and electronic applications, which places tremendous business and engineering pressures on the standard to provide solutions to the industry's challenges of cost, standardization and reuse.

The AUTOSAR standard implemented using a profile based on UML / and SysML arms engineers with a powerful tool for developing next-generation automotive systems and software.

To address these needs, the AUTOSAR initiative has developed a layered software development approach that allows companies to cooperate on standards and compete on implementations. As described in this paper, the powerful AUTOSAR standard has the promise to solve many of the engineering challenges facing electronics and software development; when implemented using a profile based on UML/and SysML engineers in the automotive market are armed with a powerful tool for developing the next generation automotive systems and software.

Bibliography

OMG SysML Specification, Object Management Group, 140 Kendrick Street, Building A Suite 300 Needham, MA 02494, USA, August 1, 2006

Technical Overview V2.0.1, AUTOSAR GbR, AUTOSAR GbR Frankfurter Ring 127 D-80807 Munich, Germany, June 27, 2006.

Unified Modeling Language: Infrastructure, Object Management Group, 140 Kendrick Street, Building A Suite 300 Needham, MA 02494, USA, May 7, 2005

Unified Modeling Language: Superstructure, Object Management Group, 140 Kendrick Street, Building A Suite 300 Needham, MA 02494, USA, August 2005

Web sites:

http://www.autosar.org

http://www.omg.org

For more information

To learn more about IBM Rational software, contact your IBM representative or IBM Business Partner, or visit:

ibm.com/software/rational

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
July 2009

RAW14150-USEN-00