2008-01-0383

Development of AUTOSAR Software Components within Model-Based Design

Guido Sandmann
Automotive Marketing Manager, EMEA – The MathWorks

Richard Thompson
Senior Engineer – The MathWorks

ABSTRACT

The steady growth in the number of electronic control units on the average vehicle and the complexity of the algorithms that reside on these controllers has resulted in one of the most significant initiatives in the automotive industry in years. AUTOSAR – the Automotive Open System Architecture – has united more than 100 companies, automobile manufacturers, suppliers and tool vendors to develop a standard architecture for electronic control units. By the end of 2006 Version 2.1 was released, and now OEMs as well as suppliers have started to develop and integrate AUTOSAR-compliant functionality and components into vehicles.

This paper will focus on the approach and challenges faced by engineers developing AUTOSAR-compliant production code using Model-Based Design.

INTRODUCTION

Version 2.1 of the AUTOSAR standard was seen by many as a maturing of the standard AUTOSAR basic software core and the stabilization of the information needed to specify application software components. This precipitated a rise in the number of tool vendors who blended the strengths of their tools with the standardized AUTOSAR platform. In 2006 Volkswagen used a model-based approach and integrated an AUTOSAR-compliant Comfort and Convenience ECU into an existing E/E environment [1].

To address the complexity of applications and algorithms automotive engineers are using Model-Based Design, which is already a widely used and accepted approach. Model-Based Design offers unambiguous and executable specifications in very early development phases. Capabilities such as automatic verification and validation and code generation are additional key benefits that make the development processes much more efficient and effective.

As a standard architecture for ECU networks, AUTOSAR is already playing a significant role in the automotive industry. Even if the current activities focus mainly on further defining and refining the standard, many OEMs and suppliers actually consider their future development processes and tool landscape taking AUTOSAR into account. This requires making decisions for a complete tool chain, starting from tools to design ECU architectures at the vehicle level down to tools for Model-Based Design modeling the functional behavior of AUTOSAR software components, as well as Run-Time Environment Generators and basic software component generation.

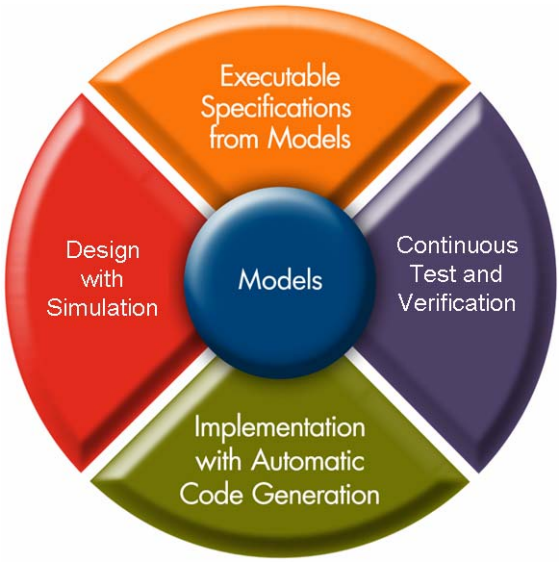

Figure 1: Model-Based Design

MODEL-BASED DESIGN

Traditional embedded software development involves paper designs and hand coding, followed by verification activities such as code inspections and a unit/integration test. Many of these activities lack tool automation, and so involve manual interaction. Thus they are error prone

and time consuming. Lack of tool chain integration provides another opportunity for errors to be injected into the software that are often detected late and at high costs to the development process.

To overcome these challenges, Model-Based Design has become a widely used and accepted approach. Not only do simulations provide insight into the dynamic and algorithmic aspects of the system, but models are also commonly used for several constructive tasks, including:

- Serving as executable specifications
- Communicating component requirements and interface definitions between customer and supplier
- Providing virtual prototypes of vehicle systems and models of drivers, road conditions, and other environmental conditions for algorithm development
- Enabling automatic code generation of the software algorithms for production systems

These steps for Model-Based Design keep the engineering process focused on error prevention and early error detection. Verification and validation early in a project can reduce the risks associated with late error detection.

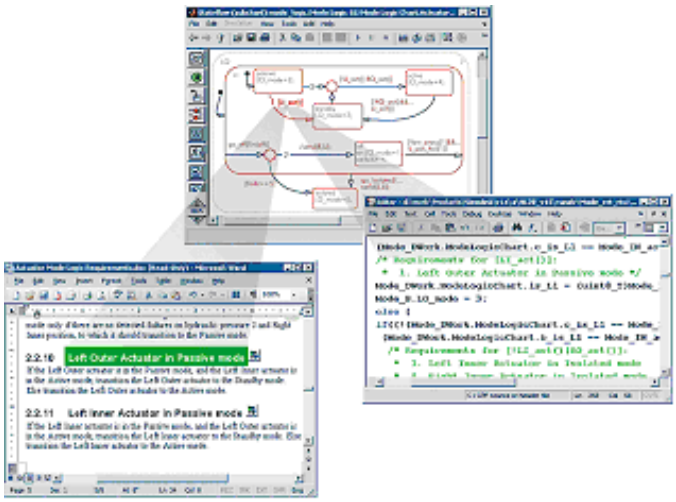

Figure 2: Requirements Traceability

As shown in Figure 1 Model-Based Design puts the model in the center of the development process enabling engineers to create executable specifications, automatically generate embedded code, and perform verification and validation (V&V) activities on these models.

REQUIREMENTS TRACEABILITY

Embedded software development using Model-Based Design typically begins with high-level system requirements. These requirements are often managed with dedicated requirements management tools, word processors, or spreadsheets. Regardless of the tool used, engineers must ensure that a model – and ultimately the generated code – fulfills these high level requirements. Hence, it is important to have a robust traceability capability that links each requirement with the corresponding part of a model. Throughout the process, engineers develop a detailed software design model and continuously perform verification and validation to ensure that the model satisfies the requirements. Models consisting of state machines and block diagrams can have links that support bi-directional traceability to the requirements defined in popular documentation and requirements management tools. Test cases – either defined by an engineer or generated automatically – can also be linked with requirements to document the appropriate coverage. In later stages of the process – after generating the production code from the implementation model for a specific target –l inks between the code statements and the model with regard to the requirements is also available. Industry standards such as IEC 61508 or Automotive SPICE™ require this high degree of traceability for safety-critical software development.

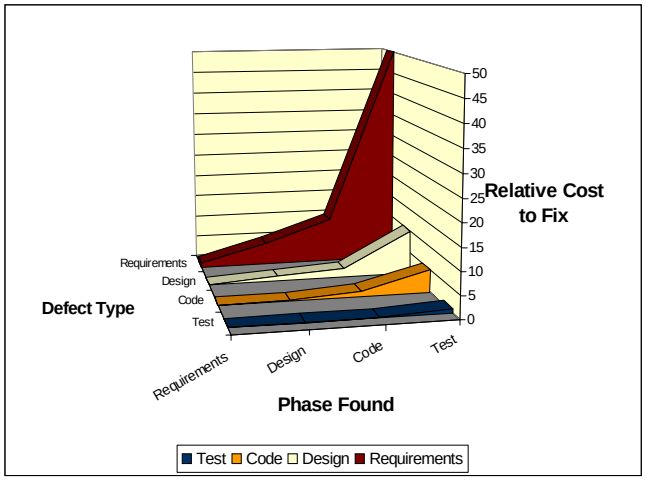

Figure 3: Relative Cost to Fix Defects per Phase Found [2]

EXECUTABLE SPECIFICATION, VERIFICATION AND VALIDATION

The requirements specified in a requirements document are then translated into a model – a formal description on a very abstract level capturing all relevant functional aspects. The primary advantages of this approach are the elimination of ambiguity and the definition of well-defined interfaces, which allow engineers to execute their concepts in the very early stages of development.

With this approach, engineers can start test and validation activities from the beginning of a project. In Model-Based Design test & validation is not just the last step in the process but a continuous verification and validation activity. In addition to test scenarios developed by a design or test engineer, there is also tool support available for automatic generation of test cases addressing coverage metrics up to MC/DC (Modified Condition/Decision Coverage).

Additionally, Model-Based Design with executable specifications facilitates communication between OEM

and supplier as well as communication between engineers on a project. Model-Based Design enables teams to complete important tasks much earlier than in a traditional approach. At the same time it leverages the quality of the complete design process, because it avoids time and cost intensive iterations if errors are found at the very end. Figure 3 shows the tremendous increase in cost for fixing errors introduced in the requirements phase but found in the test phase.

PRODUCTION CODE GENERATION

The software is automatically generated from the models during production code generation. This ensures a seamless workflow. A key benefit of this smooth integration is the reuse of the entire infrastructure that was already built and used previously for testing at the model level. If the continuous verification and validation process discovers errors, they will be fixed within the model and the software can be regenerated very quickly and efficiently.

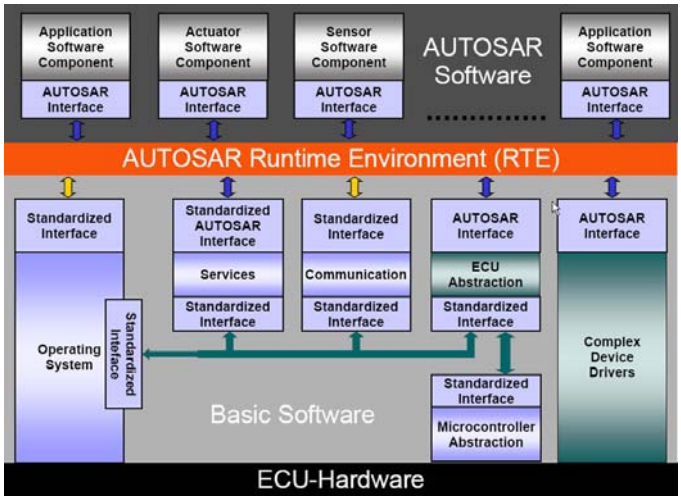

Figure 4: The AUTOSAR Software Architecture

AUTOSAR

To address the challenges of increasing complexity while developing and maintaining electronic applications within the automotive domain, the companies involved decided to define a standardized architecture for electronic control units (ECU). Therefore, in 2002 leading OEMs and suppliers established the partnership of AUTOSAR – Automotive Open System Architecture – to pursue goals including [5]:

- Implementation and standardization of basic system functions as an OEM-wide "Standard Core" solution
- Scalability to different vehicle and platform variants
- Transferability of functions throughout the network
- Integration of functional modules from multiple suppliers
- Consideration of availability and safety requirements
- Redundancy activation
- Maintainability throughout the whole product life cycle
- Increased use of commercial off-the-shelf hardware
- Software updates and upgrades over the lifetime of the vehicle

Figure 4 shows the defined architecture with the different software layers divided into three major areas:

- The Application Layer; this layer contains the functional applications of an ECU network. In the terminology of the standard they are referred to as *AUTOSAR Software Components*
- AUTOSAR Run-Time Environment (RTE): this layer was introduced to decouple the application software from the concrete infrastructure software.
- AUTOSAR Basic Software: this level between micro-controller and the RTE defines the infrastructure software both as hardware-dependent and hardware-independent non-functional services.

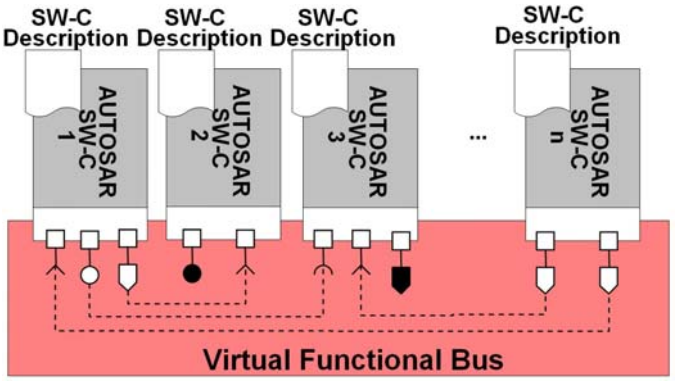

Figure 5: The Virtual Functional Bus

As mentioned above, the decoupling of functional application software and the target hardware is one of the main goals of AUTOSAR. To reach this goal, the Virtual Functional Bus was introduced. AUTOSAR software components implement the Application Layer and encapsulate the functionality of a single ECU or an ECU network. These software components have well-defined and standardized interfaces. Each AUTOSAR software component is dedicated to one specific ECU, while one ECU could consist of multiple software components.

The Virtual Functional Bus (Figure 5) implements the communication between the different AUTOSAR software components regardless of where they are located within the vehicle's ECU network. This concept in principle allows integration or transfer of AUTOSAR software components anywhere on the network. In the implementation phase the generated run-time environment is a specific instance of the Virtual Functional Bus.

DEVELOPMENT OF AUTOSAR SOFTWARE COMPONENTS

The concept and benefits of Model-Based Design are even more valuable within the context of AUTOSAR,

and should be applied to even more uses. Once a company has made the decision to develop their ECUs following an AUTOSAR-compliant process, the design or software engineer should not be forced to change his/her workflow in order to generate AUTOSAR-compliant software.

The MathWorks approach based upon MATLAB®, Simulink®, and Real-Time Workshop® Embedded Coder for generating AUTOSAR-compliant code follows a transparent and intuitive process, and it supports two different workflows top-down and bottom-up.

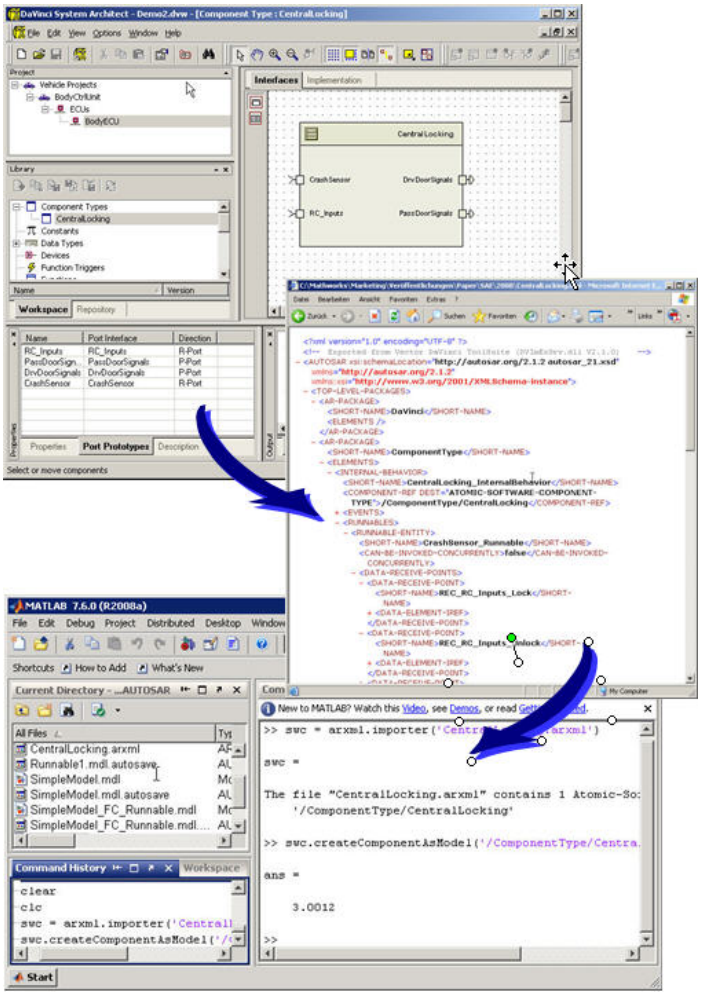

Figure 6: DaVinci Architecture Export and Import into MATLAB®

TOP-DOWN: FROM AN ARCHTECTURE MODEL TO AN AUTOSAR SOFTWARE COMPONENT

In the top-down workflow system engineers use an architecture design tool (known as an Authoring Tool in AUTOSAR terminology) such as the DaVinci Tool Suite from Vector Informatik to design an architecture of a vehicle's ECU network. The engineers then export an xml-description of the corresponding components – the AUTOSAR Software Component Description. This file contains all necessary information about the components, such as runnables, interface descriptions, data types etc. With The MathWorks AUTOSAR solution engineers can import this xml-file and automatically generate a skeleton Simulink model that contains the interface blocks (inports and outports) and the AUTOSAR-related settings for these objects as defined in the software component's description file. Figure 6 illustrates the workflow. The AUTOSAR Software Component Description file will be imported using the MATLAB command line, after which the model will then be generated. Starting from this skeleton model, design engineers can develop the controller model with Simulink, Stateflow®, and companion blocksets as shown in Figure 7.

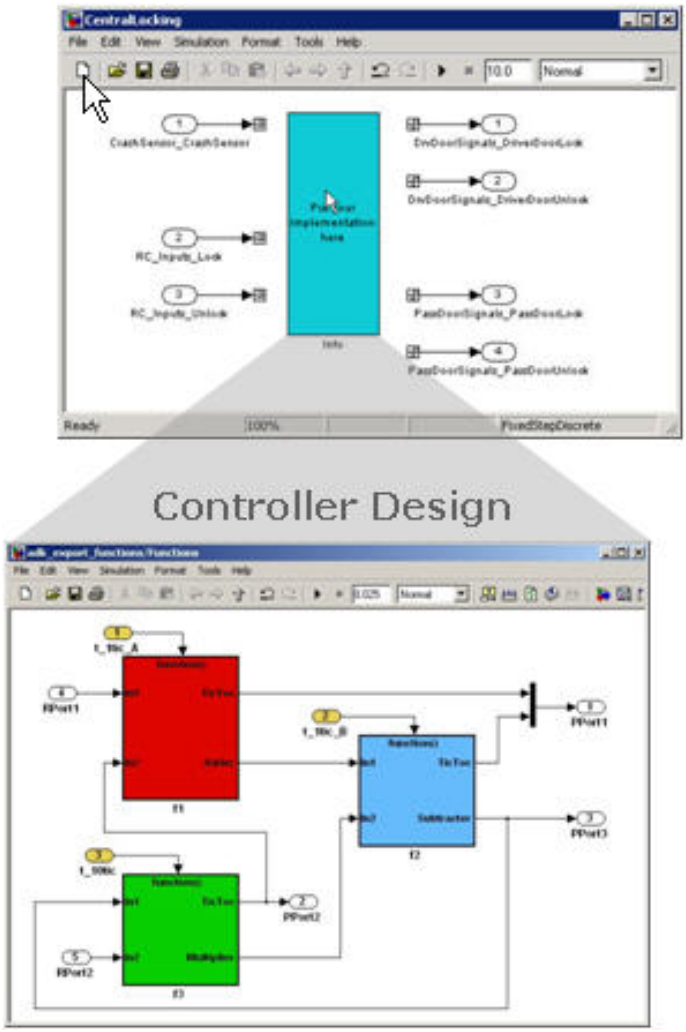

Figure 7: Controller Design

From a workflow point of view, the engineers do not need to make extensive changes. The verification and validation features especially can be applied as usual and described in the section about Model-Based Design. Typically, the design engineer makes enhancements to the model relevant to AUTOSAR – for example by introducing additional runnables or interface objects. In this particular case he/she has to make corresponding settings to ensure that the generated code is compliant to the standard and fits into the additional software environment containing the run-time environment and hardware-related components from the basic software

layer. These settings can be made via appropriate configuration parameter dialogs. For the code generation the corresponding System Target File (tlc-file) needs to be selected.

BOTTOM-UP: REUSING LEGACY MODELS TO GENERATE AUTOSAR-COMPLIANT CODE

Because Model-Based Design is a well established process in the automotive industry, companies usually have an extensive library of mature and extensively tested models. It is important that these models can be reused to target different platforms such as AUTOSAR.

The bottom-up workflow requires the same AUTOSAR configurations as described in the top-down workflow. In particular the interface objects need to be configured to ensure that the generated software component can be integrated properly. In addition to the code files, an updated AUTOSAR software component description file in XML format will also be generated automatically.

CONCLUSION

The complexity of electronic control units for automotive applications is constantly increasing, and as a consequence of that the collaboration between OEMs and suppliers led to the AUTOSAR partnership – one of the biggest standardization activities in the industry. The AUTOSAR standard is seen as the most important step towards dealing with the challenges for the development process of modern automobiles. While AUTOSAR is focusing on the software and communication aspects between AUTOSAR software components and the decoupling of the application and the Basic Software layer, Model-Based Design is an established paradigm to deal with the complexity on a functional level. Because of the different direction of both approaches, Model-Based Design and AUTOSAR are not only compatible, but they are complementary. The combination of both is an excellent way to improve the cooperation between system engineers and design engineers as well as the collaboration between OEMs and suppliers.

This paper showed that the well-established tools for Model-Based Design from The MathWorks can easily be used to develop AUTOSAR-compliant software starting from a traditional process for Model-Based Design and to enhance it smoothly by introducing the AUTOSAR-relevant aspects.

REFERENCES

1. Andreas Köhler, Volkswagen AG and Tillman Reck, Carmeq GmbH, “AUTOSAR-Compliant Functional Modeling with MATLAB®, Simulink®, Stateflow® and Real-Time Workshop® Embedded Coder of a Serial Comfort Body Controller”, MathWorks Automotive Conference 2007, Dearborn
2. NASA “Return on Investment for Independent Verification & Validation”, 2004.
3. www.autosar.org

CONTACT

Guido Sandmann
Automotive Marketing Manager EMEA
The MathWorks GmbH
Adalperostr 45
85737 Ismaning
Guido.Sandmann@mathworks.de