

AUTOSAR Model-Based Software Component Integration of Supplier Software

Saleh Mirheidari, Abdolreza Fallahi, David Zhang, and Kumar Kuppam
Navistar Inc.

ABSTRACT

AUTOSAR is finding its way into the automotive industry. European automotive manufacturing companies were the early adopters defining and promoting AUTOSAR standard. One of the main AUTOSAR goals is to improve containment of product and process complexity and risk. Increased scalability and flexibility to integrate and transfer functions is another important goal of AUTOSAR. Working with different suppliers and vendors and respect their confidentiality makes the process of application software development even more complex. Presented in this paper is a creative way of utilizing AUTOSAR to overcome the integration challenges in a multi-party object code based software integration. The run time environment (RTE) files for the application software are generated through a set of scripts to automate the process for consecutive releases. The low level device drivers are configured by one supplier and are being used by another supplier through a set of AUTOSAR client-server operation calls. The software has gone through several successful releases and is being tested in Dynamometer and Vehicle. AUTOSAR has helped establishing a common understanding of software architecture, software artifacts exchanging and integration methodology.

CITATION: Mirheidari, S., Fallahi, A., Zhang, D., and Kuppam, K., "AUTOSAR Model-Based Software Component Integration of Supplier Software," *SAE Int. J. Commer. Veh.* 8(2):2015, doi:10.4271/2015-01-2848.

INTRODUCTION

Software development has gone through major changes in the last three decades. Software is one of the fundamental factors in every product. Software development and testing has become very complex and time consuming and requires a very well-defined process. Embedded software in automotive industry is not an exception [1]. Cars and trucks are becoming more intelligent that proves the importance of software in the product. There are several sectors within automotive industry that are utilizing the software vastly. Engine control, aftertreatment control, safety airbags, anti-lock brakes, entertainment system and several other systems in vehicles are examples of sub systems utilizing software. These sub systems make the automotive industry an ideal application domain for advanced techniques from software engineering standpoint and are growing very fast year over year.

Complex software systems in vehicles include hundreds of connected hardware, such as sensor and actuators, communication networks, such as CAN and LIN (and recently Ethernet), microcontrollers with limited resources of computing power (compared to enterprise solutions) and real-time operating systems. Software plays an important role helping the manufacturers to meet tight regulations and stay in the competitive market. One can see the necessity of an efficient and effective development process for such a complex system. There are challenges in the development and deployment of software systems in automotive and commercial vehicle industry, such as speed of development, cost, maintainability and safety. As Engine Control Unit (ECU) is getting more complex according to

demand for computation, there is more than one single subsystem inside an ECU. The operating system must provide protection between software components [2].

Model-based approach is proven to be a safe, efficient and cost effective methodology for software development in automotive and commercial vehicle industry [3,5,6]. The basis for the design of a model-based methodology is a precise knowledge of targeted system class [3]. Design of the software system based on well-founded models could lead to easier analysis, validation and verification activities. Model-based design methodology has provided an environment for system design engineers to evaluate the design through simulation of the system model. This early simulation and verification helps avoiding multiple iteration of software development which reduces the development cost and improves the speed to market. Auto code generation is another key aspect of the model based design which helps to quickly turn the simulated models into software to be deployed into the target ECU.

An important step in automotive and commercial vehicle industry is to avoid proprietary solutions and to develop standard in order to simplify the understanding of the complex systems [1]. A promising step toward this direction is AUTOSAR (AUTomotive Open System ARchitecture) standard [7]. AUTOSAR project is a global partnership program to create one common standard. One of the benefits of AUTOSAR is exchangeability between supplier's solution, manufacturer's application and vehicle platform. This quality attracts automotive company's attention since it can save them time and

money in the development process. In addition to exchangeability, AUTOSAR standard addresses several other issues in the industry, such as consideration of availability and safety requirements, implementation and standardization of basic system functions as an OEM and supplier wide “Standard Core” solution etc. [8].

AUTOSAR ECU layered architecture is shown in Figure 1.

This paper will describe a creative way of utilizing AUTOSAR to integrate software components from third party supplier into the Tier 1 ECU software while respecting the suppliers' IP. This project along with its AUTOSAR implementation and integration methodology is described in following sections.

PROJECT DESCRIPTION

Emission systems are becoming more advanced to keep up with the tight regulations by EPA in North America and other agencies around the world. Aftertreatment systems show up in more products than before and control systems become more advanced and precise to achieve the desired emission levels at the tailpipe. Navistar Inc. has outsourced the aftertreatment system for its 9/10L and 13L engines. The supplier delivers aftertreatment system and control software to be integrated in Navistar products. For simplicity, we call them aftertreatment supplier.

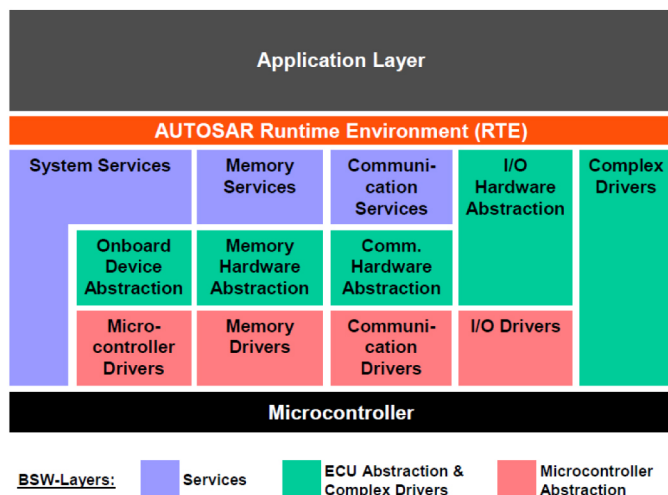


Figure 1. The AUTOSAR ECU layered architecture [8]

Navistar used the aftertreatment System from aftertreatment supplier in its 2013 products. The aftertreatment system is controlled by a standalone ECU called Aftertreatment Control Module (ACM) while the engine is controlled by the main ECU. Figure 2 shows the arrangement of engine and aftertreatment control modules. Engine control module and ACM are communicating via CAN bus.

The next phase for Navistar as part of 2017 Green House Gas program is to integrate both engine and aftertreatment control software into a single ECU. The ECU hardware is provided by another Navistar supplier. For simplicity, we call them ECU vendor. The ECU hardware and base software is delivered by ECU vendor. Navistar develops its own application software and has the

responsibility for integrating Aftertreatment supplier's software with Navistar application software. The system diagram of this phase is shown in Figure 3. The aftertreatment software is delivered to Navistar as a single AUTOSAR software component. Since this software component is integrated in the Navistar software within the main ECU, it utilizes the resources provided by the basic software such as OS services, I/O controls and interacts with Navistar's main application software. These interactions are handled through a thin layer of RTE which consists of various client server calls and implicit sender/receiver interfaces defined by AUTOSAR. The implementation of this RTE layer will be explained thoroughly in the next sections. All of the control design activities have been done in Simulink. Both Aftertreatment supplier and Navistar use AUTOSAR Target Production Package (ATPP) from Mathworks Inc. to generate AUTOSAR compliant code from models. Next section describes the steps required for building software in Navistar process.

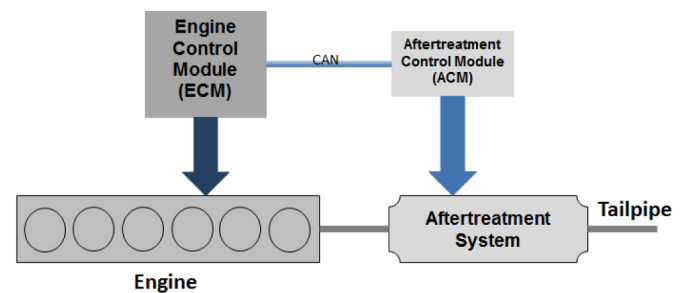


Figure 2. System diagram of engine and aftertreatment control

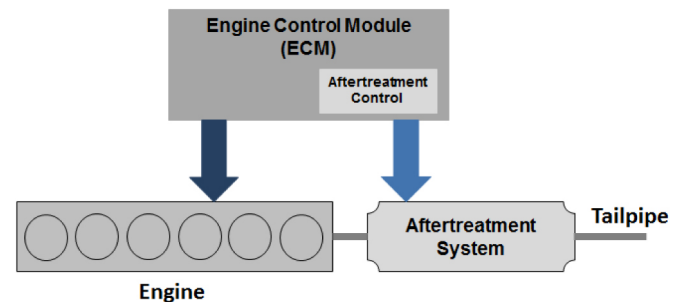


Figure 3. System diagram for engine and aftertreatment control in a single module

AUTOSAR-COMPLIANT DEVELOPMENT PROCESS

Modeling development tool and architecture design tool are two main factors in an AUTOSAR-compliant development process [9]. Simulink is an example of modeling development tool. The architecture design, RTE generation and other software configuration happen in an Authoring tool. There are different authoring tools available from different vendors and the number is growing [9]. The interaction between these two tools is shown in Figure 4. There are two distinct workflows to complete the AUTOSAR design round trip. One is the top-down workflow where the software system architecture is designed in Authoring tools and the component descriptions are exported from the authoring tool in the format of arxml files. These arxml files can be imported into the design tools