Development Process for AUTOSAR-based Embedded System

Kisoon Sung and Taeman Han

Electronics and Telecommunications Research Institute

{kssung, tmhan}@etri.re.kr

Abstract

Recently automotive embedded system has developed highly since the advent of smart car, electric card and so on. They have various value-added system for example IPA (Intelligent Parking Assistance), BSW (Blind Spot Warning), LDWS (Lane Departure warning System), LKS(Lane Keeping System)-these are ADAS (Advanced Driver Assistance Systems). AUTOSAR (AUTomotive Open System Architecture) is the most notable industrial standard for developing automotive embedded software. AUTOSAR is a partnership of automotive manufacturers and suppliers working together to develop and establish an open industry standard for automotive E/E architectures. In this paper, we will introduce AUTOSAR briefly and demonstrate the result of automotive software - LDWS (Lane Detection & Warning System) - development.

Keywords: *AUTOSAR, Layered Architecture, ECU, Simulink, ADAS, LDWS*

1. Introduction

Recently automotive embedded system has developed highly since the advent of smart car, electric card and so on. They have various value-added system for example IPA (Intelligent Parking Assistance), BSW (Blind Spot Warning)m LDWS(Lane Departure warning System), LKS (Lane Keeping System)-these are ADAS (Advanced Driver Assistance Systems). AUTOSAR (AUTomotive Open System Architecture) is the most notable industrial standard for developing automotive embedded software. AUTOSAR is a partnership of automotive manufacturers and suppliers working together to develop and establish an open industry standard for automotive E/E architectures. AUTOSAR aims to improve complexity management of integrated E/E architectures through increased reuse and exchangeability of SW modules between OEMs and suppliers. Moreover AUTOSAR intends to improve flexibility for product modification, upgrade and update, reliability of E/E system, and to enable detection of errors in early design phase. To achieve that, AUTOSAR defines their solutions that are

- Standardization of specification exchange format,
- Provision of Basic Software,
- Microcontroller abstraction to exchange without need for adaptations of higher software layer,
- Runtime Environment to encapsulate functions creates independence of communication technology,
- Standardization of interfaces to improve reusability of modules cross OEM and exchangeability of components from difference supplier.

AUTOSAR standardizes open software architecture, methodology and application interface to reflect these solutions.

AUTOSAR architecture part suggests the open software architecture that includes a complete basic software stack for ECUs as an integration platform for hardware independent software applications. AUTOSAR methodology part defines exchange formats and description templates to enable a seamless configuration process of the basic software stack and the integration of application software in ECUs. And it defines the methodology how to use this framework as well [1].

In this paper, we will introduce AUTOSAR briefly and demonstrate the result of automotive software, LDWS (Lane Detection & Warning System) development.

This paper is organized as follows. In next section, we will introduce AUTOSAR architecture and methodology. We will describe the AUTOSAR software code composition as well. In section three, we will describe the development procedure LDWS software. Finally, in section four, we draw some conclusion.

2. AUTOSAR Overview

Figure 1 shows the AUTOSAR software architecture that distinguishes four basic layer, application, RTE (RunTime Environment), BSW (Basic SoftWare) and microcontroller. The RTE provides the infrastructure services that enable communication to occur between AUTOSAR software-components as well as acting as the means by which AUTOSAR software-components access basic software modules including the OS and communication service. [2] The BSWs are divided into functional groups that are System, Memory, Diagnostic and Communication Services. These services are used in automotive system commonly.

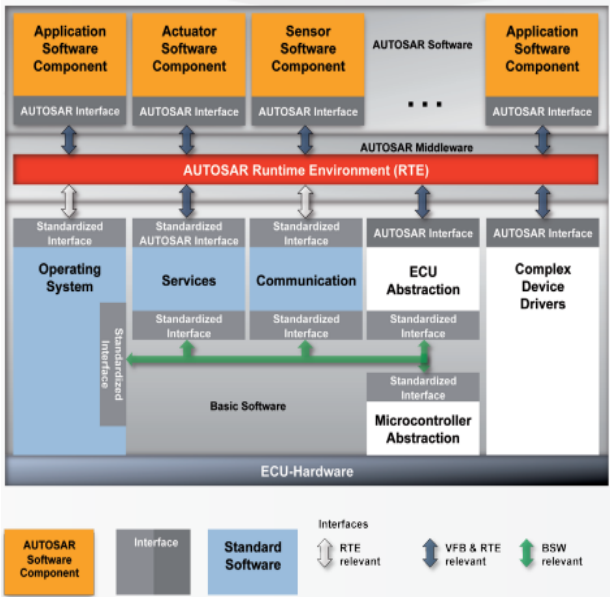

Figure 1. AUTOSAR Software Architecture

The AUTOSAR methodology is foreseen to support activities descriptions and use of tools in AUTOSAR. The AUTOSAR methodology is not a complete process description but rather a common technical approach for some steps of system development. It defines activities and

work-product except role and responsibility. AUTOSAR meta-model defines the contents of the work product and is a formal description of all the information that is produced or consumed in the AUTOSAR methodology.

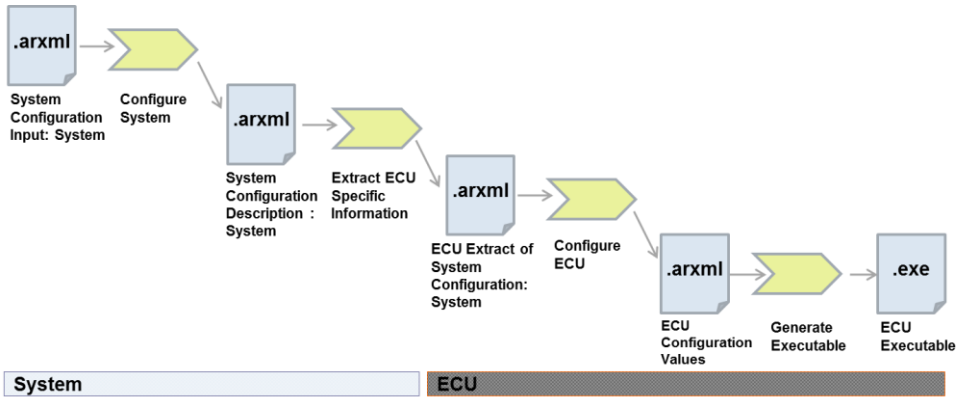

Figure 2. AUTOSAR Methodology [3]

Figure 2 shows the outline of AUTOSAR methodology. In each step, AUTOSAR-based software is developed as follow procedure.

Step1

1) Define the item to development, requirement, and constraints

2) Describe SW-Component independently of hardware (SW Component description)

3) Describe hardware independently of Application Software (ECU Resource description)

4) Describe System – network topology, communication (System description)

Step2

1) Distribute SW-Component description to ECU

Step3

1) Generate required configuration for AUTOSAR infrastructure per ECU (ECU configuration description)

Step4

1) Generate software executable based on configuration information for each ECU (Executable code)

In General, it is supported by tool though all over the development step, for example Tresos by EB, SystemDesk by dSpace, and Davinci by Vector.

On the other hand, AUTOSAR-based software code is implemented in different step. Figure 3 shows the structure of software code. Application skeleton code is generated by AUTOSAR support tool based on SW Component description in step 4. Application logic code is SW component's internal code executing specific function. It is made by general purpose code generation tool, for example Simulink by Mathworks, or by hand. It is out of scope of AUTOSAR. RTE code is generated code based on RTE configuration information using AUTOSAR support tool in step 4. Configuration code in BSW (Basic SoftWare) is the generated code based on BSW configuration information by AUTOSAR support tool in step

4, too. BSW core code is made prior to ECU integration and provided by AUTOSAR support tool generally. Microcontroller driver's core code is provided by chip vendor.

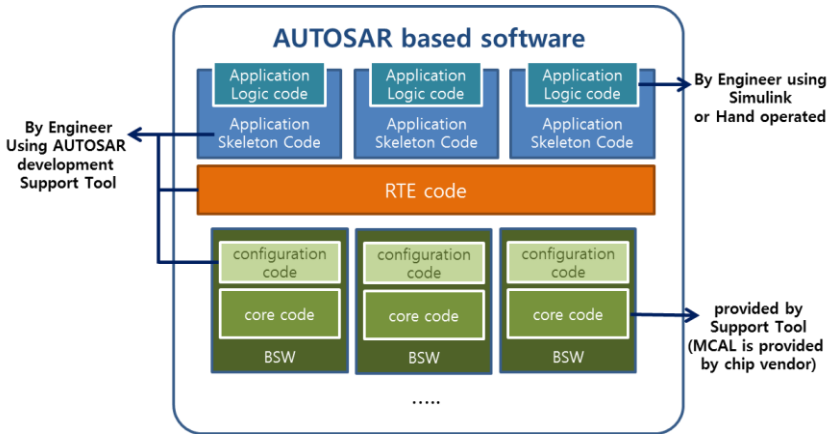

Figure 3. The composition of AUTOSAR software code

3. Development AUTOSAR based Embedded System

In this section, we demonstrate the development of a LDWS controller according to AUTOSAR methodology.

First, we set up the development environment. Table 1 shows the tool chain to develop LDWS controller. And our target hareware is Infenion XC-2300 chip.

Table 1. Tool chain to develop LDWS controller

Tool		Role
AutoWorks	ETRI	System and ECU configuration
Simulink	Mathworks	Model simulation, Application logic generation
CANoe	Vector	CAN simulation in HIL test environment
Trace 32	Lauterbach	Debug in HIL test environment

LDWS has two main functions, the detection of lane departure and the decision of warning. The first function is realized by AUTOSAR CDD (Complex Device Drivers) and notice to the decision component after detection of lane depature. The second function is realized AUTOSAR SW-Component and decide if warn to drive or not according to direction of departure and turn signal.

3.1 System Configuration Step

In this step, we describe the procedure and result of the LDWS system configuration according to Step1 in chpter2.2. In addition, we show the model simulation and application logic generation activities using Simulink as well.

First, we design software architecture complying with the SW component and the AUTOSAR standardized interface meta-model. And then we configure internal behavior of SW component. Internal behavior is designed by creating runnable entity, triggering event, point of data in/out, *etc.* Runnable entitie is the smallest code-fragment and a subject for scheduling by OS. Figure 4 is the LDWS software architecture in Autoworks and Figure 5 is the internal connection among the SW component.

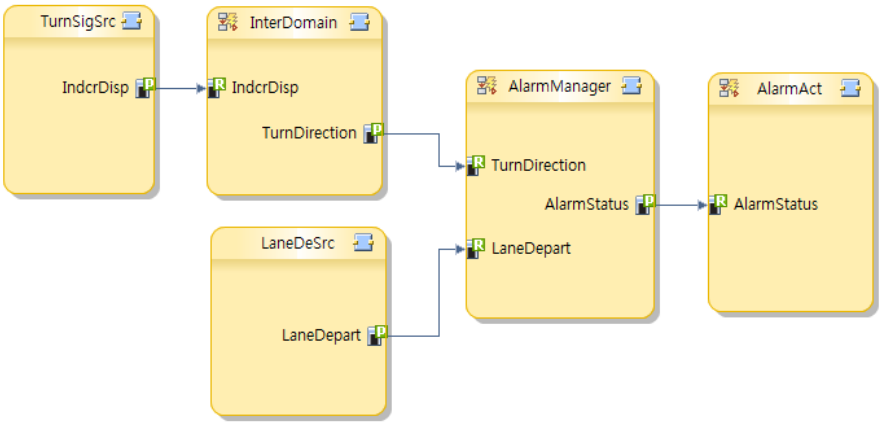

Figure 4. The LDWS software architecture in Autoworks

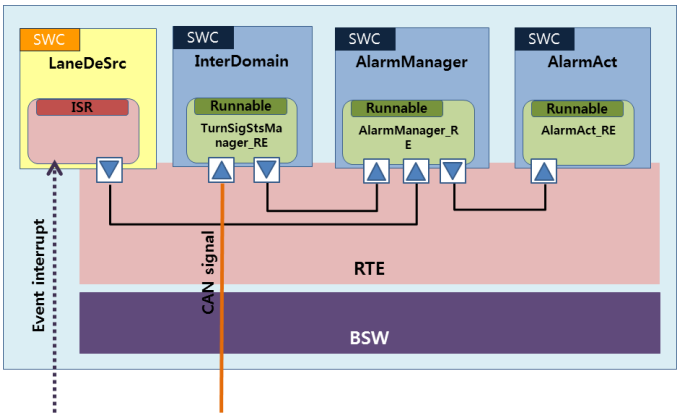

Figure 5. Interconnection of SW component

Table 2. The function description of SW components

SW component	Description
LaneDeSrc	Detect lane departure and notice to AlarmManager
InterDomain	Get turn signal from other ECU, Turn signal is transferred via CAN
AlarmManager	Decide to send warning signal to the driver by comparing departure direction and Turn signal
AlarmAct	Start to send warning signal to the driver, for example shaking the wheel or sound

After complete software architecture, we design hardware topology and communication matrix. LDWS is composed of three ECUs, lane detector, turn signal detector and warning controller. And then, we configure AUTOSAR CAN communication model, Signal-Pdu-Frame.

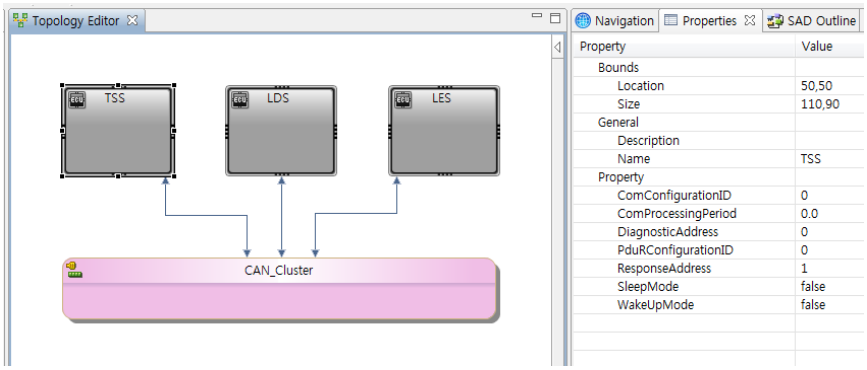

Figure 6. Hardware Topology in Autoworks

After all configurations, we map the SWC to ECU, DataElement of AUTOSAR interface to real signal. The result of the system configuration step is SW component description, hardware resource description and system description.

To simulate on MIL level, we use the Simulink that support AUTOSAR software development. This process is out of scope of AUTOSAR. We can import SW component description (.arxml) in matlab command window and Simulink translates AUTOSAR meta-model to Simulink library. We use state-flow to simulate the LDWS function.

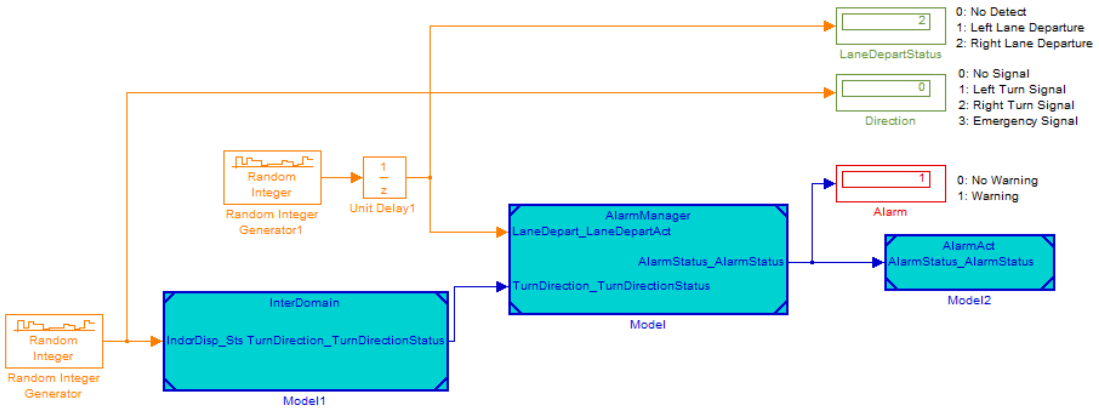

Figure. 1. Simulink model of LDWS

Figure 7 is the Simulink model of LDWS. Using AUTOSAR package of Simulink, we can verify software architecture and generate the application logic code.

3.2 ECU Configuration Step

In this step, we describe ECU configuration activities using Autoworks according to Step3, 4 in chpter2.2.

LDWS include OS, SchM, EcuM, Com, PduR, CanIf, CanTrcv, CanDriver, Gpt, Mcu, Dio, Port and RTE. We edit relative parameters using Autoworks and the result is the ECU configuration description.

OS and RTE configuration is one of the most important part in ECU configuration phase because it is related with all over the scheduling attributes. In OS configuration time, we create tasks and configure priority, system stack size and others for each

created task. And then, in RTE configuration time, we map runnable entity to task and configure the runnable entity's position in task and OS event related with runnable entity. In mapping time, we consider the kind of event for runnable entity because that will affect the performance. For example, runnable entity with waitpoint has to be mapped to its own task. Because mapping multiple runnable entities with waitpoint to the same task can lead to big delay times if e.g. a waitpoint is resolved by the incoming event, but the task is still waiting at a different waitpoint.

Lastly BSW configuration code and RTE code is generated based on the ECU configuration description. RTE code includes application skeleton and OS taskbody, all API for interfaces and data access. BSW configuration code include variable code that dependent on configuration information. Figure 8 show the distribution on ECU of LDWS.

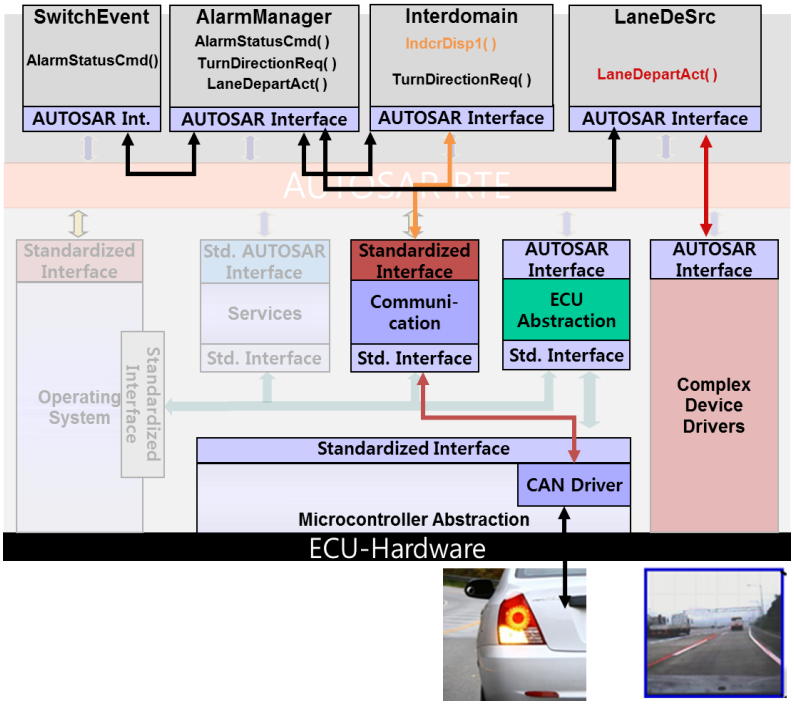

Figure 8. Distribution on ECU of LDWS

3.2 HIL Test

Test is the out of scope of AUTOSAR. But is is the normal process of an automotive application system development.

Our HIL test environment consists of CANoe, Trace32 debugger and the ECU containing AUTOSAR-based LDWS application. CANoe simulate the environment of the ECU by genrating turn signal. Figure 9 shows the HiL test environment of the LDWS system, developed following the AUTOSAR methodology.

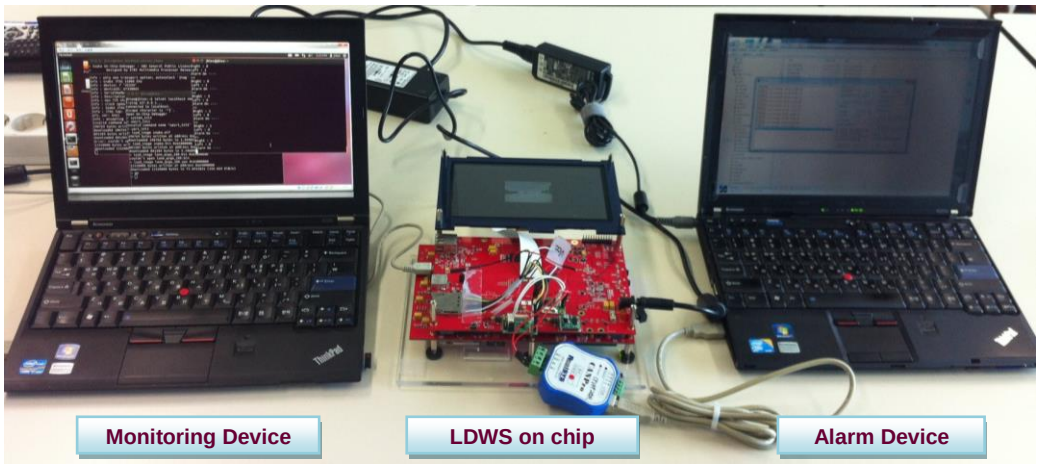

Figure 9. The HIL test environment

4. Conclusion

In this study, we demonstrated all over the procedure of AUTOSAR application development. We showed the overview of AUTOSAR architecture and methodology, and described each step of development AUTOSAR application.

AUTOSAR prepares standardized workflow, format and template to raise the reusability and reliability of the automotive software. So, the benefit of adopting AUTOSAR is the reduction of the period for developing new application by reusing application interfaces and BSW core functions. In our study, we reused some application interface provided by AUTOSAR and 50% of BSW parameter configuration that derived from prior project. Consequently we can develop the application in half the time of a prior application. This is rough estimation but accurate estimation of software development effort needs to consider numerous factors in software projects such as project size, programming language, application type, team size, development platform and so on [5].

AUTOSAR has to compensate its standard in the view of functional safety. Lately automotive functional safety is a hot issue in vehicle industry. According to ISO26262-automotive functional safety standard, AUTOSAR methodology is the part of "Development at the software level". ISO26262 suggests various feature of software, for example, freedom of interference among software component or time critical operation and function. AUTOSAR must add new features to methodology and template to qualify ISO26262 for AUTOSAR based-application.

The emergence of new technologies is permitting to collect big data. [6] Car has varied information too, and they are connected driver's safety directly. In other domain, for example, mobile device, there is security test methodology to insure the functional security [7]. So AUTOSAR must consider that point to upgrade their methodology.

Finally, LDWS application in this study is the project of research purpose, so it's difference from real production more or less.

Acknowledgements

This work was supported by the Industrial Strategic Technology Development Program funded by the Ministry of Knowledge Economy (MKE, Korea) [10041648, Robust Fault-Resilient SW for Vehicle Processors].

References

[1] AUTOSAR_GuidedTour, http://www.autosar.org.
[2] AUTOSAR_SWS_RTE R3.1, http://www.autosar.org.
[3] AUTOSAR Methodology R3.1, http://www.autosar.org.
[4] AUTOSAR SoftwareComponentTemplate R3.1, http://www.autosar.org.
[5] V. Khatibi, D. N. A. Jawawi and E. Khatibi, "Investigating the Effect of Using Methodology on Development Effort in Software Projects", International Journal of Software Engineering and Its Applications, vol. 6, no. 2, **(2012)** April.
[6] K. -j. Kim, S. -p. Hong and J. Y. Kim, "A Study of Privacy Protection from Risk of Hijacking Data", International Journal of Multimedia and Ubiquitous Engineering, vol. 8, no. 1, **(2013)** January.
[7] K. Rhee, H. Kim and H. Y. Na, "Security Test Methodology for an Agent of a Mobile Device", International Journal of Security and Its Applications, vol. 6, no. 2, **(2012)** April.

Authors

Kisoon Sung

Kisoon Sung reveived her B.S degree and M.S degree in Information & Communication Engineering from Chungnam National University. She has joined in Electronics and Telecommunications Research Institute since 2001. She researches automotive embedded systems and develops AUTOSAR development tool. Her research interests are embedded system and software engineering.

Taeman Han

Taeman Han received his B.S degree in Electronic Engineering from Kyungpook National University and M.S in Computer Engineering from Chungnam National University. Currently he complete his Ph.D. course at Chungnam National University.

He leads the national project to develop/ commercialized the AUTOSAR platform and tool. Her research interests are embedded system and software engineering, especially AUTOSAR and ISO26262.