SmartSAR: A Component-based Hierarchy Software Platform for Automotive Electronics

Hong Li, Pan Lu, Min Yao, Nan Li
College of Computer Science, Zhejiang University
Hangzhou 310027, Zhejiang, China
{lilong, lvp, myao, linan}@zju.edu.cn

Abstract—As the automotive electronic system grows larger and more complex, research on software architecture and development methodology becomes more important. This paper proposes a component-based hierarchical software platform for automotive electronics - SmartSAR, which follows the AUTOSAR specification and provides a series of tools for model-driven development, visual configuration and automatic generation. The case study with SmartAMT, a control system of automated mechanical transmission, has shown the advantage of SmartSAR platform in automotive electronic software design.

Keywords- AUTOSAR; component; Automotive Electronics; Software Platform

I. INTRODUCTION

Software technology is the core technology of automotive electronics. As the automotive electronic control system grows with more functions and more complicated structure, complexity in software structure and workload in software design involved in the automotive electronics is also increasing vastly. By 2010, premium class vehicles are expected to contain one gigabyte of on-board software [1]. However, the current software development process in automotive electronics industry follows a “vertical-mode integration” style developed by individual enterprise, which makes automotive system software lack of reusability and compatibility. Moreover, this leads to low reliability.

It will be very meaningful to conduct research on automotive electronic basic software platform compatible with international standards, referring to the developing trend of international standardization of automotive electronics, changing the developing style to “distribution-mode integration”. It will radically reduce the product development time and reduce the costs, improve reliability of products so as to enhance the level of automobile electronics industry and increase the core competitiveness of the corporations.

International AUTOSAR (AUTomotive Open System ARchitecture) organization raised the structure of layered horizontal automotive software architecture, which increases the reusability and portability of automotive electronic software by abstracting hardware, making basic software and application software into component [2]. It represents the direction of automotive electronic technique future.

SmartSAR is a software platform for automotive electronics developed by the Embedded System and Engineering Center in Zhejiang University. Following the international AUTOSAR specification, SmartSAR platform makes a clear layered view of automotive electronics software, which separates application software and the underlying software completely by the means of loose coupling mechanism of components and software run-time environment. Meanwhile, SmartSAR has taken the advantage of MDA (Model-Driven Architecture) design in current vertical-mode development [3], extending the model conversion to make it well combined with the AUTOSAR distributed-mode configuration. It has provided the automotive electronic software development with a complete solution from requirement modeling, application component design and visual configuration to automatic generation, which will reduce the development cost and increase the quality of product.

The remainder of this paper is organized as follows. In the next section, we shall present related work such as AUTOSAR solution and MDA tools. Then we describe the architecture and implementation of SmartSAR in Section 3. In Section 4, we offer a case study to describe an example of AMT (Automated Mechanical Transmission) using this platform. Finally, Section 5 summarizes this paper and advises the future work.

II. RELATED WORK

In this section, we will describe the background knowledge of SmartSAR, AUTOSAR and Model-Driven development. Then we show the advantage of SmartSAR.

A. AUTOSAR

AUTOSAR is an open and standardized automotive software architecture, jointly developed by BMW, Bosch, Continental, DaimlerChrysler, Ford, Opel, PSA Peugeot Citroen, Siemens, Toyota, Volkswagen and other major car manufacturers as well as electronic systems, semiconductors and the software supplier in 2002 [2]. AUTOSAR came up with the idea of “Cooperate on standards, compete on implementation”. It is a cooperative organization of car manufacturer and provider, and has actually set up an open industrial standard for automotive electronic software and hardware architecture.

The AUTOSAR partnership aims to establish a standard which serves as a platform for future vehicle applications. It focuses on establishing the basic software architecture. With the well-defined standardized basic software, they can bridge
the gap between microcontroller hardware and application software. The AUTOSAR partnership has defined a set of methodologies, supporting distributed and function-driven development process, standardized each ECU’s (Electronic Control Unit) software architecture in the system and defined the compatible interface between software components at the application level. It will finally resolve the problem of software getting more complicated and increase the flexibility of product modification and updating, which finally increases the quality and reliability of products.

The technical concept of AUTOSAR methodology is a layered model, which is innovative in automobile electronic application software design. With the concept of VFB (Virtual Functional Bus), it is possible to separate application and basic software. An application consists of interconnected software components, and VFB provides these components with communication mechanism and services. The VFB view of AUTOSAR is shown in the figure 1 [2].

The concept of VFB implemented on a specific ECU leads to the concept of RTE, which means Runtime Environment, just like a middleware of embedded system. As shown in the figure 2 [2], the layered architecture of AUTOSAR ECU contains two basic layers: application layer and basic software layer. The two parts are connected by RTE. In principle, this layered model is applicable for almost all the automotive applications.

AUTOSAR standard has already been preliminary attempted in some automotive software manufacturer. Vector has provided AUTOSAR solutions, including AUTOSAR development process compatible tool chains - Davinci and the embedded codes conform to AUTOSAR standard - MICROAS [4]. EB has also provided EB tresos studio, which are AUTOSAR compatible configuration, verification, generation tools and tresos AutoCore, which is a middleware followed AUTOSAR standard [5].

B. Model-Driven development

Model-Driven software development is a software engineering approach developed from the 1990s. It adopts model as the universal description in all stages of the software development. It guides the stages of requirement analysis, design, verification, code generation and so on with models. Model-Driven development ensures the consistency in all stages with the automated tools, which improves the development efficiency and quality [6].

Research on Model-Driven development includes research on MDA, UML and the tools supporting Model-Driven design.

MDA is a methodology proposed by OMG. The nature of MDA is to use models to guide design and maintenance of systems. Through the conversion between models, MDA is able to fulfill the automated process from requirements to codes.

UML is developed on the basis of Booch, OMT and OOSE approaches and it has absorbed the advantage of many object-oriented modeling approaches. Soon after it was developed, UML was accepted by OMG as one of their standards. UML unifies the basic concepts, terminologies and approaches of object-oriented modeling, provides the common language for software developers, which is very useful as an industry standard and has been widely adopted for modeling a variety of software systems.

Many institutes and companies have developed tools that support Model-Driven development. Telelogic Rhapsody is a Model-Driven IDE based on UML and SysML, and it is widely used in embedded system software project development [7]. Developed by Esterel Technologies, SCADE is a high-level security embedded application development environment referring to the fields of aviation, national defense, energy and so on, which has covered the whole process in embedded software development from requirements to codes [8].

C. Advantage of SmartSAR

Following the AUTOSAR standard, SmartSAR platform implements the separation of application and basic software by layered architecture and loose coupling mechanism of components. SmartSAR uses XML document and
standardizes interface as unified format for data exchange. SmartSAR inherits MDA method and makes an automatic model conversion from requirement model to system configuration file and domain models for application SWCs (software components).

Current AUTOSAR solution like Vector and EB’s product are all mainly focused on the configuration in accordance to AUTOSAR process, lacking related tool on requirement modeling and component generation. However, SmartSAR platform has well inherited the advantage from MDA vertical-mode development, combined with AUTOSAR distribution-mode configuration, to provide the users with complete solution.

III. ARCHITECTURE AND IMPLEMENTATION

A. SmartSAR Overview

SmartSAR, as an automotive electronics basic software platform, can be divided into two major parts: the running environment and the development environment. The former one includes OS (operating system), BSW (basic software), RTE (runtime environment) and HAL (Hardware Abstraction Layer). The latter one is an IDE (integrated development environment), which consists of Model-Driven design and conversion tool chains, visual automated configuration tool chains and automatic generation tool chains.

The overall architecture of SmartSAR platform is shown in figure3.

![Architecture of SmartSAR](image)

The functions of the parts are summarized as follows.

- SmartSAR OS provides functions which conform to the OSEK OS specification to RTE, such as task scheduling, resource management, event management, interrupt management, and so on.
- SmartSAR BSW provides communication services to RTE.
- SmartSAR RTE allocates and maps application SWCs to ECUs and supports the communication between SWCs.
- SmartSAR HAL lies on the bottom of the platform, enclosing details and discrepancy of microcontroller and ECU to the upper layers.
- SmartSAR IDE includes modeling, conversion, configuration, generation and so on.

The detail of each part is described below.

B. SmartSAR OS

SmartSAR OS is a real-time operating system, based on the SmartOSEK OS [9] [10] [11]. SmartSAR OS includes the following major function modules:

- Task scheduling management. The module is in charge of task activation, task termination, the conversion between states and scheduling based on priority. SmartSAR OS supports three schedule strategies of tasks: non preemptive scheduling, full preemptive scheduling and mixed preemptive scheduling.
- Event management. It implements functions to wait for events, set and clear event mask for synchronized requirements between extended tasks.
- Alarm management. Here the alarm is based on system clock, which will be triggered if the counter has reached a given value. When alarm occurs, it can activate a task, a callback function, or set event.
- Resource management. This module is used to coordinate the resource sharing between tasks or interrupts, which ensures that two tasks or interrupts do not acquire the same resource at the same time.
- Interrupt management. It supports two categories of ISR (Interrupt Service Routine). ISR category 1 does not use an operating system call, which will not lead to task switch. It will execute faster and occupy less resource. ISR category 2 has lower priority, but it can use operating system calls.
- Stack monitor. It monitors the stack in the process of task context switches without memory management unit. The module makes it possible for early detection of stack overflow error and handles it properly.
- Protection mechanism. This module monitors the task execution time and the time blocked by lower priority tasks, to ensure that the task execution deadline is met.
- Exception management. This module provides exception management and debugging tools with hook functions for users to customize inner execution of exceptions.

All the SmartSAR OS functions are provided to RTE in the form of system services.

C. SmartSAR BSW

According to AUTOSAR Specification, automotive electronics basic software includes communication services, memory services, diagnostic services, and so on. In SmartSAR BSW, we have implemented the communication module so far.
Communication service is a module used for vehicle network communication, which provides unified interface for communications between different application components and unified services for network management, and hides lower layer protocol and driver details. SmartSAR BSW mainly supports three kinds of bus protocol services: CAN, LIN and FlexRay.

SmartSAR BSW communication services provide two communication modes: S-R (Sender-Receiver) mode and C-S (Client-Server) mode. S-R communication involves the transmission and reception of signals consisting of atomic data elements that are sent by one component and received by one or more components. C-S communication involves two entities, the client or user of a service and the server that provides the service. The server, in the form of the RTE, waits for incoming communication requests from a client, performs the requested service and dispatches a response to the client.

For inter-ECU communication, the RTE uses the functions provided by SmartSAR BSW communication services. For intra-ECU communication, the RTE can use the same services, or implement the functionality on its own if it is more efficient.

D. SmartSAR RTE

RTE is the implementation of the concept of VFB on a specific ECU. SmartSAR RTE lies between application SWCs and OS, as well as BSW. It is an important interactive environment between them. For RTE hiding implementation details of OS and BSW, application SWCs are independent from specific platforms.

Designers of application SWCs only need to define their components’ communication interface with other components in accordance with norms, and then implement the functions. Designers of the system will layout the functions needed to implement on the level of system, and choose the application SWCs used by these functions, and then map them to the ECUs. After that they can use the automated tools to configure and generate the executable codes for each ECU. The process of SmartSAR RTE is shown in figure 4. It takes the following steps:

1. SmartSAR RTE analyze system architecture description, component communication network topology description and application SWCs interface descriptions to synthesize system configuration description.
2. Extract the ECU allocation description according to the system configuration description and get the initial configuration description for each ECU.
3. Configure OS and BSW information according to the requirement by each ECU, finish the allocation of OS resource and mapping of application SWCs to task, regenerate detailed ECU configuration description and generate RTE files for ECU.
4. According to the ECU configuration description, RTE files, application SWCs implementation files, OS source files (or library files) and BSW source files (or library files) are finally compiled to the executable file on ECU which can be downloaded to object board.

E. SmartSAR HAL

HAL reduces the dependency between OS, BSW and hardware, increases the reusability and portability of OS and BSW. SmartSAR HAL consists of the MCU abstraction layer and the ECU abstraction layer, as shown in the figure 5.

MCU abstraction layer provides drivers to control the device, such as drivers for internal EEPROM, internal CAN controller and internal ADC. ECU abstraction layer abstracts the physical characteristics of ECU on-board devices, provides unified interface to OS and BSW. The current version of SmartSAR HAL has included hardware abstraction packages for HCS12 and MPC555.

F. SmartSAR IDE

SmartSAR IDE establishes the integrated development environment for designing and implementing automotive electronics software, based on eclipse platform that provides
plug-in mode of functional components development and integration. SmartSAR IDE consists of three parts: the Model-Driven development and conversion tool SmartSAR Designer, the visual configuration tool SmartSAR Configurer and the automatic generation tool SmartSAR Generator.

SmartSAR Designer starts from the UML requirement modeling and it can automatically export system architecture configuration XML description file through model conversion. Meanwhile, it extracts the application SWCs information, converts it to SmartC domain model for detailed design automatically [12], and then generates its XML description file and source file for future configuration and compilation. At last, the component files will be saved in system resource library for reuse.

SmartSAR Configurer uses the system configuration XML description file and the SWC XML description file to configure the system. It contains several tools for network configuration, RTE mapping and allocation, ECU configuration (such as OS configuration and BSW configuration), and so on.

SmartSAR Generator will generate codes that conform to MISRA C standards and documents based on model arguments. The SmartSAR will invoke the corresponding ECU compiler to get the object file. Documents in Word format will be automatically generated, which greatly reduces workload of writing and maintenance of documents in the process of software development.

The figure 6 illustrates SmartSAR IDE tool chains.

![SmartSAR IDE tool chains](image)

SmartSAR has fully taken advantage of MDA and well linked with industrial mainstream development method. It provides complete solution for transition to AUTOSAR development method. With these automated tools, SmartSAR IDE has reduced the error rate drawn into by designers in traditional code development style, increased reliability of automotive electronic software and efficiency of development.

IV. CASE STUDY

SmartAMT (Smart Automated Mechanical Transmission) is a control system, which is developed by the Embedded System and Engineering Center in Zhejiang University. It is based on traditional manual gear transmission. The software control system will enforce the process of vehicle shifting automatically according to the driver’s intention and state of vehicle. SmartAMT version 1.0 was completed in 2006, it has been to on QQ EZDriver Car of Chery with the existing executable body and replacing the original Marelli ECU with own hardware/software [13]. SmartAMT 2.0 follows the design ideas of version 1.0, using SmartSAR platform and makes improvement in development for software.

A. Experimentation

In our experiment, we use 16-bit microcontroller HCS12 for ECU of AMT control system, and use 32-bit microcontroller MPC555 for ECU of engine control system. SmartAMT application is divided into 10 SWCs. SWC A to SWC I are all distributed to ECU of AMT control system and SWC J to ECU of engine control system:

- **SWC A**—Handle position judging component
- **SWC B**—Automatic gear choosing component
- **SWC C**—Switch gear execution component
- **SWC D**—Clutch separation component
- **SWC E**—Clutch combination component
- **SWC F**—Oil pressure adjusting component
- **SWC G**—Meter display component
- **SWC H**—Engine protection component
- **SWC I**—Data monitor component
- **SWC J**—Engine coordination component

Combination of SWC B, C, D, E and J are the key parts of AMT system. It decides whether gear switch is needed according to input from vehicle speed, rotate speed and throttle change. When switch is needed, it enforces the engine coordination, clutch control and switch execution. SWC D, E and I must get the respond in time and should be executed non-preemptively, so they have the highest priority in the application. SWC B and C have the normal priority.

SWC H monitors the engine rotation speed. When the speed drops sharply, SWC H will control the clutch so as to protect the engine. Considering the engineers’ experience, SWC H should rank second in priority that is below that of clutch control and engine coordination.

SWC F, G and I are routine operations including adding oil pump pressure, meter display and data monitor. They have the lowest priority and execute in cycle.

SWC A will judge the handle position to make sure the car’s situation. The priority of this component should be the same with SWC B and C.

Those SWCs which have the same priority are mapped to one task on special ECU. Then we invoked CodeWarrior compiler for HCS12 and MPC555 to get the executable files. The deployment process of SmartAMT is shown in Figure 7.
B. Result discussions

All 10 SWCs have been designed by SmartSAR Designer. Combined with OS, BSW, HAL and RTE on SmartSAR platform, they have been configured rapidly by SmartSAR Configurer and generated executable file automatically through SmartSAR Generator. The modeling interface of clutch control component is shown in the figure 8.

As shown in the Figure 9, the combination of clutch (Curve2) can be divided into three stages: idle motion stage, semi-combined stage and full-combined stage.

On the stage of idle motion, the clutch combines rapidly and for the control of SWC I, engine rotation speed (Curve4) increases smoothly.

At the beginning of semi-combined stage, the clutch processes slowly and the engine rotation speed decreases slowly. This time, the car speed (Curve3) changes slightly. With the clutch combination degree increase, engine rotation speed increases slowly after the decrease because the pedal action increases the throttle open degree (Curve1), and the car speed also increases slowly.

At the stage of full-combination, rotation speed and car speed have gotten conformance and now rapidly combine transmission which can lead to less combination time.

From the experiment data, we find that rotation speed curve changes smoothly and the change rate of car speed is not obvious, which means that the sample car starts up with a small degree of impact.

SmartAMT 2.0 adopts new software architecture and the MDA development method, takes advantage of automatic tools in developing process. Compared with the developing and debugging process of version 1.0, at least half of the time is saved. Meanwhile, the 10 independent SWCs have been saved in our system resource library as reusable resource, which lays a good foundation for future research and development.

V. CONCLUSION AND FUTURE WORK

This paper borrows idea from international AUTOSAR software architecture, and brings forth a component-based layered software platform SmartSAR, which contributes to the reduction of coupling between application software and basic software/hardware. Corresponding IDE has provided complete tool chains for automotive electronics software distributed assembly, which ultimately increases the efficiency and quality of software product development. The example of SmartAMT has shown the advantage of SmartSAR.

Figure 7. The deployment of SmartAMT.

Figure 8. Modeling interface

In the experiment on the QQ EZDriver Car, data monitor component sends data of clutch position, engine rotation speed, car speed and throttle pedal position through CAN bus periodically. On PC end, we use a self-made CAN tool to receive, display and analyze the data. For the performance of the starting up at first gear is a critical fact on evaluating the transmission, we choose the critical data curse at first gear condition as the analytical object here.
AUTOSAR is a large software architecture which is still under development. SmartSAR is just an initial instantiation and there are many future works:

- Reliability of OS (such as introducing error avoidance method and error detection mechanism).
- Completeness of BSW (such as completing diagnostic services and memory services).
- Improvement on RTE scheduling and mapping algorithm.
- Completeness of IDE tool chains.

ACKNOWLEDGMENT

This research was supported by 863 National High Technology Program under Grant No. 2007AA01Z129 and 2007AA010304 and Key Program of Science and Technology Department of Zhejiang Province under Grant No. 2008C01036-3.

REFERENCES