Experiences with AUTOSAR compliant Autocode generation using TargetLink

Naveen Alwandi, Manjunath BC
Delphi Electronics & Safety

ABSTRACT

Increased safety, comfort and emission norms are pushing the complexity of vehicle systems up exponentially. Model-based development processes have increasingly been adopted for the development of automotive embedded control software to help implement the complex systems and reduce the development time. Model-based and autocode technology has become mature and brings many advantages in automotive software development.

In parallel, consortium of major OEMs and suppliers are driving towards standard specification of automotive software architecture, AUTOSAR (AUTomotive Open System Architecture). AUTOSAR would enable flexibility for product modification, upgrade and update scalability of solutions within and across product lines.

To model algorithms and generate AUTOSAR compatible Autocode has become the necessity for the projects using AUTOSAR architecture. To streamline development phases and shorten them effectively, seamless integration between the architecture system design models and algorithm models, ECU testing and calibration is required. This paper focuses on the coupling between architecture system design models and algorithm models and to Autocoding.

MAIN SECTION

Transition to Modeling and Autocoding:

Over the years, embedded system is characterized by increasing complexity, tedious development cycles, high quality at lower cost and quicker time to market. As the complexity of algorithms increased, software (SW) development has transitioned to higher level of abstraction so that the developers can develop and maintain the algorithms with better and easier understanding. Over decades the transition from low level language to high level language enabled the engineers to concentrate more on algorithm to be developed and less on the implementation details. This abstraction also meant that C code could be reused on different hardware simply by using different target compilers.

As the complexity continued to increase, there was no other choice but to move one level higher in the abstraction layer which is nothing but Model based development. It is well known in the industry that model development offers various advantages such as ability to simulate designs upfront, support for rapid prototyping, code generation, which enable not only shorten the development cycle but also improve the product quality.

Software Architecture and AUTOSAR

Model based development has been an enabling factor to meet the demands of growing complexity and faster development time but it primarily address only to the Application Layer of the Software. Software development for the lower layers like Communication, Diagnostics, actuator drivers etc which are more close to the hardware are generally not suggested to be modeled and autocoded as they are more tightly coupled with hardware and involve more register configurations and relatively less algorithm or logical implementations. Suppliers generally use layered SW architecture with reusable building blocks for the lower layers. Software with clearly defined interfaces enables independent development of the modules, reuse of SW and reduction in testing effort. But since different suppliers used their customized software architecture, the effective reuse from OEM perspective was less. Specifically it did not allow OEMs to migrate a feature from one ECU to another ECU easily as the interfaces between the application layer and the lower layers were not defined in an industry wide standard approach. Also change in microcontroller still required a considerable amount of effort for supplier.

Leading OEMs and Tier 1 suppliers, having recognized this to be an industry-wide challenge, decided to work together to address it. AUTOSAR (AUTomotive Open System ARchitecture) is a worldwide development partnership of car manufacturers, suppliers and other companies from the electronics, semiconductor and software industry. Common objective is to create a basis for industry collaboration on basic functions while providing a platform which continues to encourage competition on innovative functions. Goals of the architecture includes 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 network, integration of functional modules from multiple suppliers etc. The AUTOSAR standard will serve as a platform upon which future vehicle applications will be implemented and will also serve to minimize the current barriers between functional domains. It will, therefore, be possible to map functions and functional networks to different control nodes in the system, almost independently from the associated hardware. The standardized architecture, detailed specification for the lower layers and standardized component description files and interfaces provides the flexibility to move the features across nodes and also to change the hardware.

**Brief Overview of AUTOSAR**

To achieve modularity, scalability, transferability and re-usability of functions AUTOSAR will provide a common software infrastructure for automotive systems of all vehicle domains based on standardized interfaces for the different layers shown in the image below. The division is basically based on dependency on HW. SW feature functions which are hardware independent and hence reusable are termed Software Components and the hardware dependent parts are termed as Basic Software.

![AUTOSAR Architecture Diagram](image)

**AUTOSAR architecture**

The ECU software is primarily partitioned into the following three parts: 1) Software Component (SWC): The AUTOSAR software is the functional part of the ECU software, for example, the controller code. It consists of software components that are hardware-abstracted. All software components together form the application layer. 2) AUTOSAR run-time environment (RTE): RTE is the middleware software that allows ECU function development independently of the ECU hardware. The RTE is ECU-specific. Each ECU has an RTE of its own. 3) Basic Software (BSW): The basic software contains hardware-dependent parts of the software as well as the operating system, communication drivers, and AUTOSAR services.

The AUTOSAR Software Components encapsulate an application which runs on the AUTOSAR infrastructure. The AUTOSAR Software Components have well-defined interfaces, which are described and standardized within AUTOSAR. For the interfaces as well as other aspects needed for the integration of the AUTOSAR Software Components, AUTOSAR provides a standard description format, i.e. the Software Component Description (SWC-D). With the standardized interface definition and SWCs being hardware-abstracted they can thus be transferred from one ECU to another in a network consisting of several different ECUs. Communication between the SWCs and SWC to the lower layers is all through the RTE. Hence SWCs become independent of each other and can thus be transferred from one ECU to another without rearranging other SWCs.

The RTE manages communication within the ECU software – in the application layer, and between the application layer and the basic software. At system design level, (i.e. when drafting a logical view of the entire system irrespective of hardware) the AUTOSAR Runtime Environment (RTE) acts as a communication center for inter- and intra-ECU information exchange. The RTE provides a communication abstraction to AUTOSAR Software Components...
attached to it by providing the same interface and services whether inter-ECU communication channels are used (such as CAN, LIN etc.) or communication stays intra-ECU. RTE is generated using RTE generator tools.

Basic Software is the standardized software layer, which provides services to the AUTOSAR Software Components and is necessary to run the functional part of the software. The basic software comprises all hardware-specific ECU software. The AUTOSAR standard describes the interfaces that the basic software uses to provide services and access to data of sensors and actuators connected to the ECU. It consists of the operating system (OS), the communication layer (COM), and other service-oriented software such as NVRAM, Flash and memory management, Diagnostics etc. It also includes Microcontroller Abstraction Access to the hardware which is routed through the Microcontroller Abstraction layer (MCAL) to avoid direct access to microcontroller registers from higher-level software. MCAL is a hardware specific layer that ensures a standard interface to the components of the Basic Software. It manages the microcontroller peripherals like DIO, ADC, SPI etc and provides the components of the Basic Software with microcontroller independent values. BSW modules, MCAL libraries specific to the microcontroller along with MCAL, OS, COM generator tools to generate configuration options are used to build Basic Software.

Software Architecture Modeling Tools

As the interfaces and methodology is standardized, many architecture tools such as DaVinci from Vector, SystemDesk from dSPACE and Picea from Mecel are available to develop software architecture and system model. Architecture tools allow defining and integrating SWCs. Several SWCs can be combined to form a software architecture that can be used as a part of an overall system model. Architecture Tools enable to maintain and track the growing number of functions, and high level of networking, that handling several hundred software modules.
DaVinci Developer – Design tool for AUTOSAR-conformant ECUs

SWC definition in the architecture model includes definition of the port interfaces, data types, port prototypes with service needs and communication specification, definition of runnable entities with activation events and port access etc. Tools can be flexibly used in a distributed development process based on the AUTOSAR method as the information associated with a SWC can be imported and exported in the System Description Template format (AUTOSAR standardized XML formats). This enables tight coupling of the interfaces in all phases of development and thereby reducing the bugs, and reduce re-work due to incorrect interfaces or types of the interfaces.

Key attributes of SWC Description

**SW Component Description**

- General characteristics (name, manufacturer, etc.)
- Communication properties:
  - _p_ ports
  - _r_ ports
  - interfaces
- inner structure (composition)
  - sub-components
  - connections
- required HW resources:
  - processing time
  - scheduling
  - memory (size, type, etc.)

Development of SWC as Models

As mentioned earlier modeling and autocoding is generally suggested for Application layer. So algorithm modeling tools like Simulink and Stateflow from Mathworks can be used to model SWCs. AUTOSAR compliant C code can be generated using tools like TargetLink(TL) from dSPACE, which provide AUTOSAR specific blocks to specify AUTOSAR interface attributes.
There are different development approaches to develop the AUTOSAR compliant models. Broadly they are divided based on whether software component's behavior, i.e., the controller algorithm is already available or the software component's description is already available. Top-down approach is used where the AUTOSAR data for a SWC is available in System Description Template format. Typically this approach is used if AUTOSAR is deployed at the start of the project itself. Conversely, when migrating a non-AUTOSAR project to AUTOSAR, then bottom-Up approach is used when software component's description is not available and algorithm model is available.

In the top-down approach, when designing the AUTOSAR software of an ECU or ECU network, software component descriptions can be specified before actually implementing the software components. And the individual SWC description can be exported into XML file format. SWCD file can be imported into TargetLink's DataDictionary (DD) which is a container of all SWC attributes. Using this SWCD file a frame/skeleton model can as well be generated. If the algorithm model is available then it can be inserted into skeleton model to make an implementation model from which code can be generated. Alternatively, implementation model can be built manually and SWC attributes can be referenced from TL DD manually. This method of retaining the old models, importing SWC xml file only into the DD and updating the existing algorithm model for the changes in runnables, interfaces etc is generally found to be more simpler and cleaner approach though it takes more effort especially for projects doing incremental development.

In Bottom – Up approach, AUTOSAR data in the form of software component descriptions is not available, so TargetLink DataDictionary itself can be used to build the SWC description of the component. It involves creating software components, creating runnables and interfaces at a high level. Different RTE trigger events like
TIMING_EVENT Trigger (periodic), DATA_RECEIVED_EVENT Trigger etc can be defined and assigned to a Runnable. Different scalar, array and struct data types can be defined to be associated with the data elements of the interfaces.

Bottom-Up approach: Example of DataDictionary with different Objects and Attributes to be populated

Bottom-up approach: SWC-D exported from TL DD into xml and imported to Software Architecture Tools

SWC Model to Code

Once the DD is prepared with the details of the SWC description, AUTOSAR attributes have to be referenced in the TargetLink AUTOSAR blocks for code generation. For instance, a runnable created under software component object in the Data Dictionary should be referenced in the model for an atomic subsystem with a Runnable block that implements the functional behavior. TargetLink generates one C function for each runnable as required by the AUTOSAR standard.
Mapping of the Runnable under SWC object in DD to the Function block in Model

Generated code uses and defines the appropriate API calls as per AUTOSAR standard. To simulate the behavior of the RTE, TargetLink generates an excellent simulation frame, which maps the RTE macro or function calls in the SWC to global variables and also generates access functions to global variables. This frame acts as a dummy replacement to the RTE code (generated by RTE generator tools for the entire ECU) and enables quick testing of the SWC on the host PC in SIL mode or PIL mode without any need of the developer to create any stubs.

```c
void demoRunnable(void)
{
    ... /* Some controller/ControllerRunnable/1
    # combined # call of function: controller/ControllerRunnable/Rte_Function1 */
    # combined # call of function: controller/ControllerRunnable/Rte_Function1 */
    S1 = S2 = 0;
    S2 = 0;
    S1 = S2 = 0;
}
```

TargetLink generated code snapshot showing RTE calls

SWC description details under the component can be exported from TargetLink DataDictionary as AUTOSAR XML file for integration with architecture tool or reuse in other projects.

Challenges and Issues:

In this paper we share our experiences of deploying AUTOSAR within model based development and in particular with AUTOSAR design tool DaVinci, Matlab, Simulink and Stateflow modeling tools and production code generator dSPACE TargetLink. There is a seamless interaction between the TargetLink and the DaVinci. However there were some minor issues and challenges faced. Some of the limitations and scope for future improvements are discussed below. These issues or limitations are with reference to the versions of the tools used. DaVinci Developer and TargetLink interact by exchanging AUTOSAR XML files in accordance with AUTOSAR methods. This top-down procedure is used for the initial design and also for subsequent development iterations. A well-coordinated AUTOSAR tool chain greatly simplifies the model-based development of AUTOSAR-compliant software.

Simulation of application errors RTE APIs (CS and SR for to pass error codes):

TL currently doesn’t support simulation of application error signals from RTE APIs. For instance, SWC has a fallback behavior when a CAN signal is outdated. When modeling this, the status signals from RUNNABLE INPORT block can be read and implemented. However, simulating COM signal outdated callback in Matlab is not possible and hence the fallback behavior cannot be simulated early within Matlab environment itself. On similar lines server calls
cannot be simulated with the same model within Matlab environment and alternative solutions have to be used to ensure correctness of implementation.

**ARXML and TL DD incompatibilities:**

In the top-down approach we had little nigling issues with populating TL DD with DaVinci generated SWC-D. One such issue was with names of scaling and datatypes that they cannot be same. Another such issue was Period parameter for initializations were needed to be changed from 0 to -1 for inherited sample time.

**Multiple instantiation of SWC and handling PIM:**

Conceptually, multiple instantiation of SWC is similar to libraries, and used when same functional behavior is to be executed multiple times. For instance, “Door Control” could be one such feature function which is then instantiated four times for each door in a car. Multiple instantiation is very much used especially in ECUs realizing Body Control features such as door control, mirror control, exterior light controls. Multiple instantiation is not yet supported by TargetLink but will hopefully be supported in future.

**Using RTE support files for simulation instead of creating TL’s own framework:**

It would be a nice feature if for SIL and possibly for PIL simulation RTE files themselves are used instead of TL generating its own framework. That would bring the testing of SWC and its interfaces in Matlab environment closer to real code testing.

**CONCLUSION**

With advent of AUTOSAR it has become necessary for all players to adapt the development processes. Even though AUTOSAR has standardized methodology, interfaces, exchange formats, there are lot of challenges for successful deployment. OEMs will now have to define the vehicle architectures in more strict way, while the suppliers have to align their development process and tools to meet AUTOSAR challenges. Tool vendors have the challenges of making tools interoperable so that they work seamlessly across all the development phases. There is a seamless interaction between the production code generator dSPACE TargetLink and the AUTOSAR design tool DaVinci Developer. Given the fact that dSPACE has system architecture software and RTE generation tool in its shelf, it is best positioned to make a seamless integration of behavioral and architectural code generators which is the need of the hour.

**ACKNOWLEDGMENTS**

The authors would like to acknowledge and thank the teams in Delphi Technical Center India; Delphi Deutschland, Bomig; CTC, Kokomo; for all the contributions and guidance.

**REFERENCES**

1. [http://www.dSPACE.com](http://www.dSPACE.com)
SAE World Congress, Detroit 2007-01-0507

CONTACT
naveen.alwandi@delphi.com
bc.manjunath@delphi.com