

# Implementation of ECU Configuration Framework based on AUTOSAR Methodology

Joo-Chul Lee<sup>1</sup>, Tae-Man Han<sup>1</sup>, Sang-Ha Kim<sup>2</sup>

<sup>1</sup>Electronics and Telecommunications Research Institute  
{rune, tmhan}@etri.re.kr

<sup>2</sup>Department of Computer Engineering Chungnam National University, Daejeon, Rep. of Korea  
shkim@cnu.ac.kr

**Abstract** – AUTOSAR (AUTomotive Open System ARchitecture) is a partnership of automotive manufacturers and suppliers working together to develop and establish a de-facto open industry standard for automotive E/E architectures. AUTOSAR defines software architecture, methodology, and application interfaces. The methodology describes ways to exchange formats or description templates to enable a seamless configuration process of the basic software stack and the integration of application software in ECUs and it includes even the methodology how to use this framework. The configuration process is divided into two major steps such as “System Configuration” and “ECU Configuration”. Software architecture design, hardware topology, network configuration, and system mapping are done in system configuration step. BSW (Basic SoftWare) module configuration and source code generation are done in ECU configuration step. As results of ECU configuration step, the source codes generated from ECU configuration step are merged with the BSW module sources, and they will be the complete sources for target system. This paper describes implementation of ECF (ECU Configuration Framework) for configuring parameters of BSW modules and generating source codes reflecting configuration results of BSW module parameters.

**Key words** – AUTOSAR, Methodology, Framework.

## 1. Introduction

AUTOSAR (AUTomotive Open System ARchitecture) is a partnership of automotive manufacturers and suppliers working together to develop and establish a de-facto open industry standard for automotive E/E architectures [1].

Why AUTOSAR? We can answer this question by summarizing needs from each stakeholder [2]:

- OEM
  - OEM overlapping reuse of software modules
  - Maintaining ability to compete on innovative functions, enlarged design flexibility
  - Simplification of the integration task
  - Reduction of total SW development costs
- Supplier
  - Reduction of version proliferation
  - Development partitioning among suppliers
  - Increase of efficiency in functional development

- New business models are possible
- Tool provider
  - Common interfaces with development processes
  - Seamless, manageable, task optimized (time dependent) tool landscape.
- New market entrant
  - Transparent and defined interfaces enable new business models

Each stakeholder can get above benefits from AUTOSAR.

To satisfy those requirements AUTOSAR provides three main results [2]:

- **Architecture:** Software architecture including a complete basic (environmental) software stack for an ECU as an integration platform for hardware independent SW applications
- **Methodology:** Exchange formats (templates) to enable a seamless configuration process of the basic software stack and the integration of application software in ECUs
- **Application interfaces:** Specification of application interfaces as a standard for application software modules

To produce results reflecting AUTOSAR standard AUTOSAR requires a common technical approach for some steps of system development. It is ‘AUTOSAR methodology’ that is used by AUTOSAR as the technical approach of development.

The AUTOSAR methodology is divided into two main steps like system configuration and ECU configuration. In ECU configuration step many ECU configuration parameters based on system configuration results are configured, and this configuration results are stored in XML format.

This paper shows how this ECU configuration procedure is implemented.

The rest of this paper is organized as follows: in section 2 we introduce AUTOSAR methodology; in section 3 ECU configuration metamodel is explained, and in section 4 how this ECU configuration metamodel is implemented; finally in section 5 we make conclusion.

## 2. Related works

### 2.1 AUTOSAR Methodology

AUTOSAR methodology is a kind of common technical approach for some steps of system development. AUTOSAR methodology uses SPEM (Software Process Engineering Meta-model) which is defined by the OMG (Object Management Group). SPEM is not described in this paper.

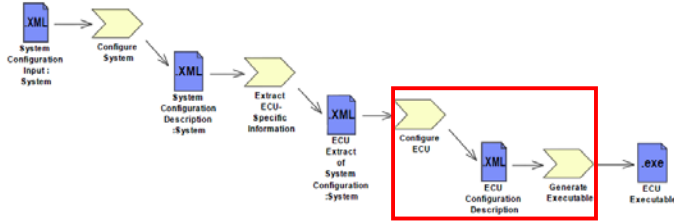


**Figure 1 Overview of AUTOSAR methodology**

AUTOSAR methodology includes design steps from the system level configuration to the generation of an ECU executable. The result of each step is delivered to the input of next step in XML format.

Firstly system configuration input has to be defined. This step is not described in Figure 1. System configuration input includes software and hardware components and identified overall system constraints. AUTOSAR provides templates to ease formal description of these initial system designs.

In configure system step, software components are mapped to the ECUs with regard to resources and timing requirements. The output of this step is “system configuration description”. This description includes all system information (e.g. bus mapping, topology) and mapping of which software component is located on which ECU.

After getting system configuration description, tool extracts ECU specific description from it.

Configure ECU activity is the scope of this paper. In this step all necessary information for implementation such as task scheduling, necessary BSW (Basic SoftWare) modules, configuration of BSW, assignment of runnable entities to tasks, etc.

In the last step, generate executables, source codes based on ECU configuration description delivered from previous step are generated, and these source codes are compiled and linked together.

### 3. ECU Configuration in AUTOSAR

Figure 2 shows ECU configuration phase of the whole AUTOSAR methodology. The major activities of this phase are extraction of ECU-specific information and configuration of the ECU, and generation of executable ECU software. Among them configuration of ECU is described in this paper.

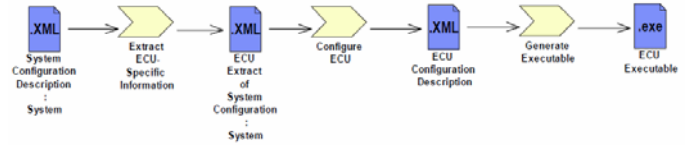


**Figure 2 ECU Configuration Phase**

### 3.1 ECU Configuration Methodology

ECU configuration is one step in the overall AUTOSAR methodology.

#### 3.1.1 Inputs to ECU configuration

ECU configuration has two input sources.

- System Configuration Description: all configurations that must be agreed across ECUs.
- BSW Module Description: information for specific BSW module. This is mainly used to provide details of the actual implementation and constraints of specific BSW module.

#### 3.1.2 ECU configuration

ECU configuration can be broken down into three activities, as show in Figure 3.

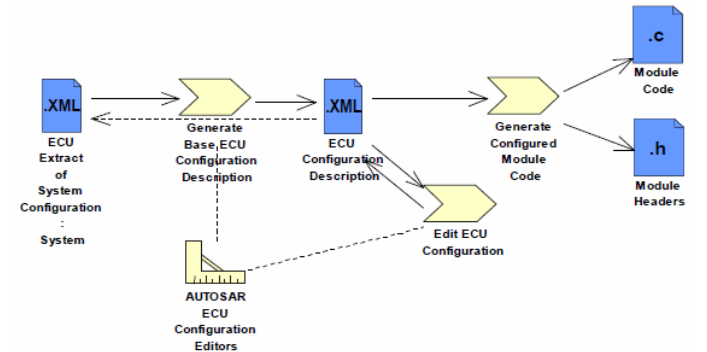


**Figure 3 ECU Configuration broken down into three subactivities**

#### 3.1.2.1 Base ECU Configuration Description

The first step in the process of ECU configuration is to generate “base ECU configuration description“. In this step SWC descriptions of application SWC prototypes, BSW service modules prototypes (such as NvRam Manager, ECU state Manager, etc) and the connections between the application SWC ports and BSW service modules’ ports are added.

Figure 4 shows Metamodel of base ECU configuration [7]. EcuSwComposition contains the complete Software Composition in an ECU, consisting both of application software components and service components (boxes in red line).

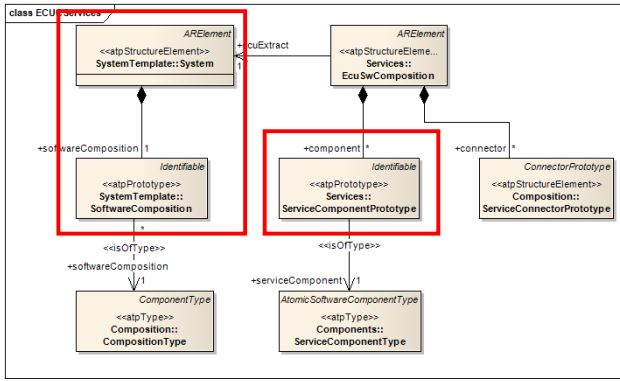


**Figure 4. Metamodel of base ECU configuration**

### 3.1.2.2 Configuration Classes

Each Configuration parameter for BSW modules has its time to bind. AUTOSAR supports three bind times such as pre-compile time, link time, and post-build time.

The configuration in different process-steps has some consequences for handling of ECU configuration parameters. For example, if a configuration parameter is defined as pre-compile time, after compilation this configuration parameter cannot be changed any more.

The configuration class of a parameter is typically not fixed in the standardized parameter definition since several variants are possible (Variant is a kind of preset which is comprised of configuration class value for each configuration parameter). However once the module is implemented the configuration class for each of the parameters is fixed in that implementation. Choosing right configuration class from the available variants is depending on the type of application and the design decisions taken by module implementer.

### 3.1.2.3 Edit ECU Configuration

The second step in the process of ECU configuration is to edit the configuration parameters for all BSW modules. In this step ECU configuration description could be updated iteratively (see Figure 5.).

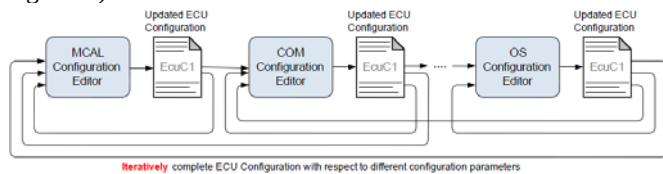


**Figure 5. Iterative use of ECU configuration description**

Iterations will be divided between several organizations due to the fact that parameters within a BSW module are either configured pre-compile time, link time or post-build time. Typically pre-compile time parameters are configured by a Tier2 supplier and post-build time parameters are configured by the OEM (Original Equipment Manufacturer). Link-time parameters can either be configured by a Tier1 or Tier2 supplier.

The binding time of each parameter could affect the other parameter. A pre-compile time parameter can affect a post-build time parameter. The methodology supports description of dependencies between parameters (BSWMD does this role). BSWMD is another input for building ECU configuration description. BSWMD contains a description of the entire BSW module, including dependencies between parameters. Each con-

figuration parameter has its affection value. These descriptions of dependencies make it possible to inform about changes that will affect other organizations taking part in ECU configuration.

## 3.2 ECU Configuration Metamodel

AUTOSAR exchange formats are specified using a Metamodel based approach. The Metamodel for the configuration of ECU artifacts uses a universal description language to that it is possible to specify different kinds of configuration aspects.

The configuration language uses containers and actual parameters. The containers are used to group corresponding parameters. The configuration description implementing metamodel is divided into following two parts:

- ECU configuration parameter definition
- ECU configuration description

Figure 6 shows how these are used to create configuration description.

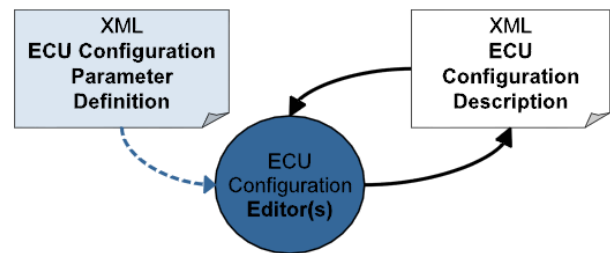


**Figure 6. ECU configuration parameter definition and ECU configuration description**

### 3.2.1 Top level structure

Figure 7 shows top level structure of ECU configuration parameter definition.

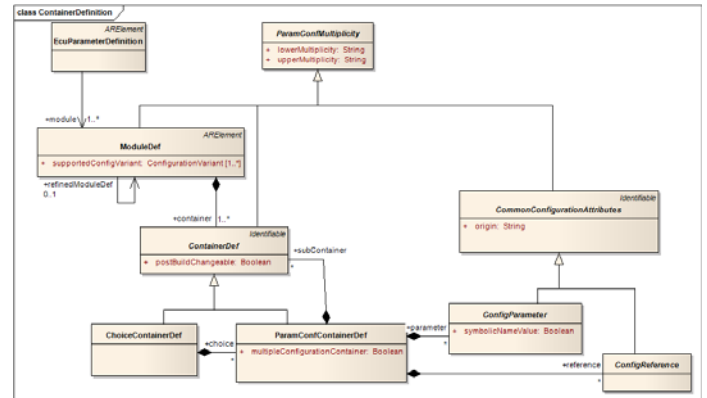


**Figure 7. ECUC (ECU Configuration) parameter definition top level structure**

ECU Parameter Definition class (EcuParameterDefinition) collects all references to individual module configuration definitions of AUTOSAR ECU configuration, and defines a reference relationship to definition of several software modules.

Module Definition class (ModuleDef) defines ECU configuration parameters of one software module such as BSW, RTE, SWC...

- supportedConfigVariant: how the configuration variants are related to the configuration classes.

### 3.2.2 ECU Configuration Description Top-Level Structure

As what we described previously, configuration description implementing Metamodel is divided into definition and corresponding description.

In Figure 8, the classes inside of dotted box are definition and the classes inside of solid line box are corresponding descriptions.

Module Configuration class (ModuleConfiguration) subsumes all configuration objects that belong to one managed software module.

- definition: it assigns ModuleConfiguration to the corresponding ModuleDef it is depending on
- implementationConfigVariant: it specifies which configuration variant has been chosen for this ModuleConfiguration
- moduleDescription: if ModuleConfiguration holds configuration values of BSW module, a reference to corresponding BswImplementation shall be provided (we can get BSWMD via this attribute)

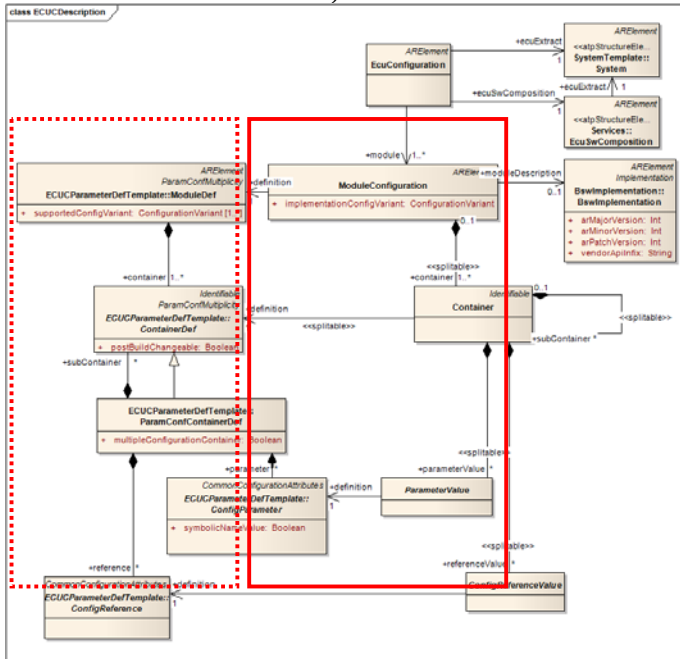


Figure 8. ECU Configuration Description Top-Level Structure

### 3.2.3 Container and Parameter Value

Container class (Container) groups other containers, parameter values, and reference values. (ParameterValue, ConfigReferenceValue).

### 3.2.4 Parameter Definition

Parameters are defined within a ParamConfContainerDef class using an aggregation with the role name parameter at the parameter side (see Figure 9).

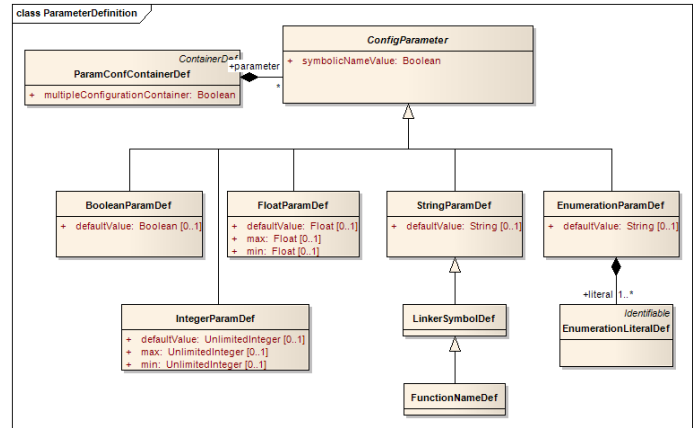


Figure 9. Parameter definition

## 4. ECU Configuration Framework

### 4.1 ECF (ECU Configuration Framework)

ECF is composed of following subcomponents (Figure 10). AutoWorks is the name of tool we are developing, it follows AUTOSAR methodology.

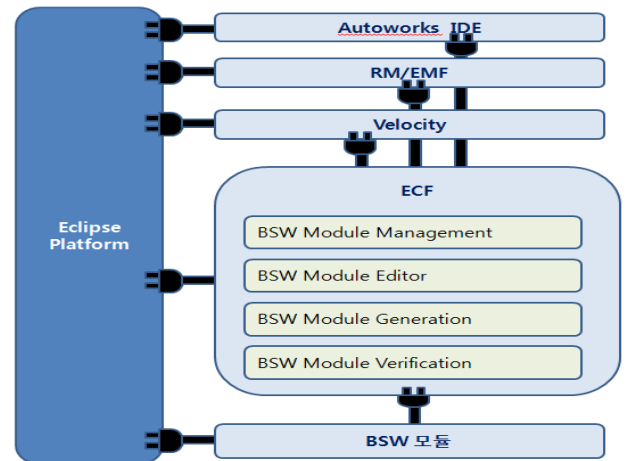


Figure 10 Overall ECF design

ECF is implemented in the form of eclipse plug-in. It provides interfaces for each function (BSW module editor, BSW module generator) to other BSW modules. It composes UI by the help of AutoWorks IDE, and it also can access model through RM/EMF.

#### 4.1.1 BSW Model Management

ECF provides eclipse extension points to the BSW modules. Each BSW module describes extension point, and therefore it can be looked up by user. Users can select BSW module from the list, and configure parameters and generate source codes by using each BSW module.

#### 4.1.2 BSW Module Editor

ECF BSW module editor function lets the user edit configuration parameters. BSW module editor function abstracts common UI and functionalities by providing *ModuleConfigurationEditor* abstract class. BSW module programmer should inherit this

class and implement module specific UI. This class also provides control class for each parameter and parameter handling class.

#### 4.1.3 BSW Module Generation

ECF BSW module generation function provides *IModuleCodeGenerator* interface. BSW module programmer should implement this interface to add source codes generation functionality. ECF adopts velocity template engine to generate variable source codes [8]. BSW module programmer should make velocity template and dispose configured parameter values in velocity template, then velocity engine merges these values with template and generate source codes.

#### 4.1.4 BSW Module Verification

This functionality provides verification interface to check integrity of parameter configuration results. Each BSW module should check integrity and report check result to ECF.

### 4.2 ECF BSW Module Editor Design

Architecture of BSW module editor module is like Figure 11.

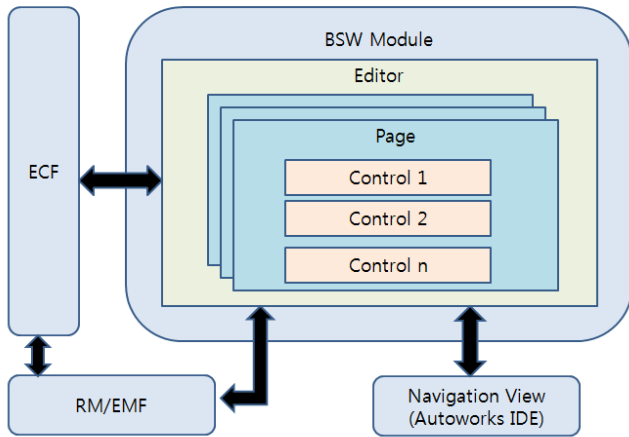


Figure 11 ECF editor architecture

BSW module plugin provides UI for editing configuration parameters. BSW module editor is composed of many configuration pages. Each page represents a parameter container in AUTOSAR meta-model, therefore it contains many parameter editing controls in it.

ECF provides editor, page, parameter control, and BSW module programmer can implement editor components by inheriting them. BSW module editor interacts with ECF to support navigation among pages and error locating. It also interacts with RM/EMF plugin to access and handle model data representing configuration parameters. Module navigation view from AutoWorks IDE plugin helps BSW module editor to navigate among pages fast. It manages containers (editor pages) as tree structure and informs BSW module programmer of each container's current position.

### 4.3 ECF BSW Source Codes Generator Design

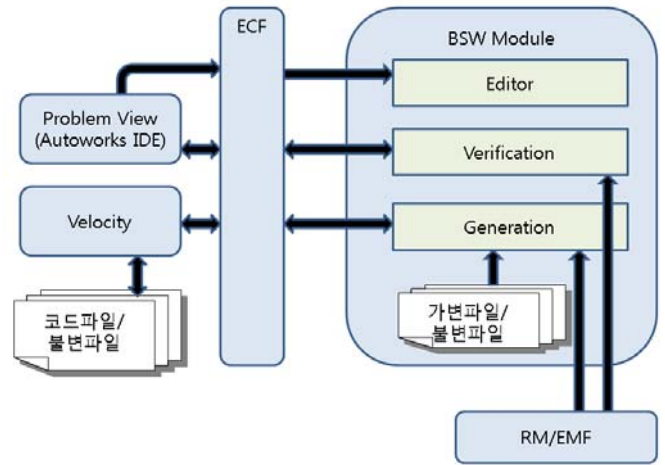


Figure 12 ECF source generator architecture

Figure 12 shows ECF BSW source codes generator architecture. BSW source codes generator uses velocity template engine plugin [8]. User selects “ECU generator” to generate source codes reflecting configuration parameter values.

At the initial phase of source generation procedure, source code generator verifies integrity of parameter configuration results. If there is any problem code generation procedure will stop and user should fix problems. If there is no problem it starts to generate source codes. The source codes generator passes variable code templates and VelocityContents data including configuration results. ECF initialize velocity engine, and generate source code by using these information. The generated variable source codes are placed output directory.

Source codes generator also passes invariable source codes for BSW modules. ECF just copies these source codes to output directory. Now user can get complete source code configured for the specific target system.

## 5. CONCLUSION

AUTOSAR requires a common technical approach for some steps of system development. This approach is called “AUTOSAR methodology”. These steps are divided into two main steps such as system configuration and ECU configuration. In ECU configuration steps many ECU configuration parameters based on system configuration results are configured and the final goals for this configuration phase is source codes configured for specific target system reflecting user's configuration parameters.

The ECF implemented ECU configuration step of AUTOSAR methodology. ECF (ECU Configuration Framework) is composed of BSW module management, BSW module editor, BSW module generation, and BSW module verification function. Through these functions ECF provides BSW module selection, parameter configuration UIs, generation of source code reflecting parameter configuration results, and verification for checking integrity of configured parameter results.

After System Configuration and ECU Configuration, user can get complete source codes set configured for specific target system.

## References

- [1] AUTOSAR technical overview 2.2.0, Nov. 2007
- [2] AUTOSAR tutorial, 1st AUTOSAR open conference, Oct. 2008
- [3] AUTOSAR Template Modeling Guide 2.2.2, Aug. 15 2008
- [4] AUTOSAR BSWMD Template 1.1.0, Aug. 06 2008
- [5] AUTOSAR ECU Configuration Template 2.1.0, Sep. 15 2008
- [6] AUTOSAR Methodology 1.2.2, Aug. 15 2008
- [7] AUTOSAR Metamodel 3.2.0, Sep. 12, 2008
- [8] Apache Velocity Project, <http://velocity.apache.org/>