Asia Oceania Systems
Engineering Conference
17-18 October 2019
Bangalore, India

Implementation of ECU Specific Requirements in ODX Generation Using Variation Management

Aswin Nata
General Motors Technical Centre India
3rd Floor, Creator building, ITPL,
Whitefield, Bangalore-560066
+919008822335
aswin.nata@gm.com

Eswar Kumar Kodati
General Motors Technical Centre India
3rd Floor, Creator building, ITPL,
Whitefield, Bangalore-560066
+919611166775
eswar.kodati@gm.com

Abstract. As the need for diagnostic applications in the advanced technological vehicles is increasing, the need for standardized format for diagnostic data is also growing. ODX (Open Diagnostic data eXchange) file is one of the standardized XML-based formats based on ISO 22901 used for exchanging diagnostic data in modern world vehicles. Different types of diagnostic identifiers like DIDs, RIDs and DTCs are defined in a structured way in ODX files. These diagnostic identifiers will be used throughout the life cycle of vehicle development like software development, testing, production, after-sales and service. The new features like Autonomous and Electrification demands to have common and distinctive diagnostics requirements for different vehicle platforms. This poses architectural challenges in managing common features and specific features with in a superset architecture (eg: Defining a single DID differently for different ECUs). One of the ways to tackle this challenge is through variation management concept. Variation management enables embedding information about different flavors of controllers into the same architecture. This helps downstream software teams to optimize their software to have their specific requirements. This paper focuses on variation management solution to accommodate ECU specific diagnostic requirements in ODX generation.

Introduction

In recent times, automotive industry is growing towards advanced technological features like autonomous and electrification etc. to meet the customer requirements. This makes us to use complex embedded systems with in the vehicles. As the complexity of the systems increases, it gives us the challenges to maintain the quality. Some of the automotive systems are safety-critical, where quality cannot be compromised. Vehicle diagnostics are becoming critical in automotive industry to confirm the desired quality. Diagnostic applications make sure the system is working as per the given requirements. This has become the integral part of the automotive vehicle development life cycle and used in different phases like development, testing and production. They also play an important role in after-sales and service. Different types of diagnostic identifiers are defined to monitor and report issues present in functionality of different subsystems of the vehicle. As the need for diagnostic applications in the advanced technological vehicles is increasing, the need for standardized format for diagnostic data is also growing. ODX file is one of the standardized XML-based formats based on ISO 22901 used for exchanging diagnostic data in modern world vehicles. Diagnostic identifiers like DIDs, RIDs and DTCs are defined in a structured way in ODX files. Every ECU in the vehicle will have its ODX file with that ECU's diagnostic information. These ODX files will be used for generation of diagnostic applications, testing the feature software during development and in-vehicle testing happening in plants and at service centers.

Diagnostic application development using models instead of manual development will improve the efficiency of the code generation. One of the widely used developmental tools used in automotive diagnostic modelling is Rhapsody. In this we model superset diagnostics data needed for different features of the vehicle and then by configuring the needed diagnostic information specific to each ECU, we generate respective ECU ODX files. As the complexity of requirements increases, the need for more diagnostic identifiers also increases. Automotive vehicles need high configuration and lots of variable requirements due to   controller differences, regulatory issues and customer requirements. This poses architectural   challenges in managing common features and specific features with in a superset architecture. For minor changes in the requirements we need re-model the complete diagnostic identifiers and then configure to the respective ECU ODX file. When considering at product line levels, defining the complete models for minor variations decreases the productivity a lot and models will become large. To avoid these issues, variation management can be used to find commonality in the variants and define the variation points for only the variation that we need. This way we can avoid the re-modelling of common artifacts and re-use the existing modes and thus reduce the manual efforts there by increasing the productivity and quality. Variation management is needed to keep in pace with the continuously increasing complexity of distributed embedded automotive systems. The capability of effectively applying variation points is very critical to manage the increasing number of varying requirements of product line development. Variation management offers benefits such as cost saving, lower redundancy, improved productivity and quality.

The current paper discusses the tools needed for variation management and process of executing variation points for the generation of ODX files for ECUs with an example.

COTS (Commercial off-the-shelf tools) for the variation management

IBM Rational Rhapsody: It provides a collaborative design, development and test environment for systems engineers and software engineers that supports UML, SysML and AUTOSAR.

Big Lever Gears: A Systems and Software Product Line Engineering Tool that lets engineer a portfolio of related products in an efficient manner, taking full advantage of the products’ similarities while respecting and managing their differences. Gears is the development environment that’s used to create, organize, and maintain your production line. Here products, profiles (configurations) and features.

Rhapsody Gears Bridge: Enables the management of product line diversity where Model behaviors for different products can be automatically configured by making feature choices in a Gears feature profile. Provides PLE extensions for SysML and UML.

Handling Variation in ODX Generation

Variation points: Gears provides a small and compact vocabulary of variation point mechanisms. When Gears does exercise a variation point, the result is a projection. A projection is the instance of the shared asset that Gears creates by exercising the variation points in that shared asset. The Gears variation point logic language is analogous to the switch or case statement in conventional programming languages. The logic statement in each variation point is a list of logic clauses. Each logic clause has a Boolean expression and an operation. When the variation point is actuated, the Boolean expression in each logic clause is evaluated, in the order of the clauses, until one of the Boolean expressions evaluates to true or until there are no more clauses to evaluate. If the boolean expression for a clause evaluates to true, the operation for that clause is executed and the evaluation of the logic statement exits. If no boolean expressions evaluate to true, then no operations are executed. Figure 1 explains how different DIDs are allocated to different products based on variation points. ODX generation captures the data accordingly.

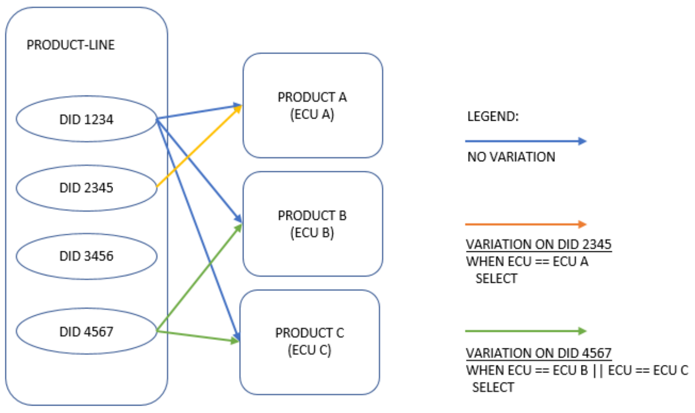


Figure 1. Generation of different products based on variation

Work Flow: ODX file is generated per ECU. The ODX files are generated by parsing diagnostics data in Rhapsody and Gears model. In the generation process, when certain model element has a variation point, the variation logic is evaluated and processed further.

1. Identify the products.

2. Define the Feature model in Gears. Figure 2 shows a sample feature model of FrontClimateControl. Based on the needs, respective atoms (Atom: the lowest possible selection) can be selected. E.g.: For front control climate system, either Auto or Manual can be chosen.

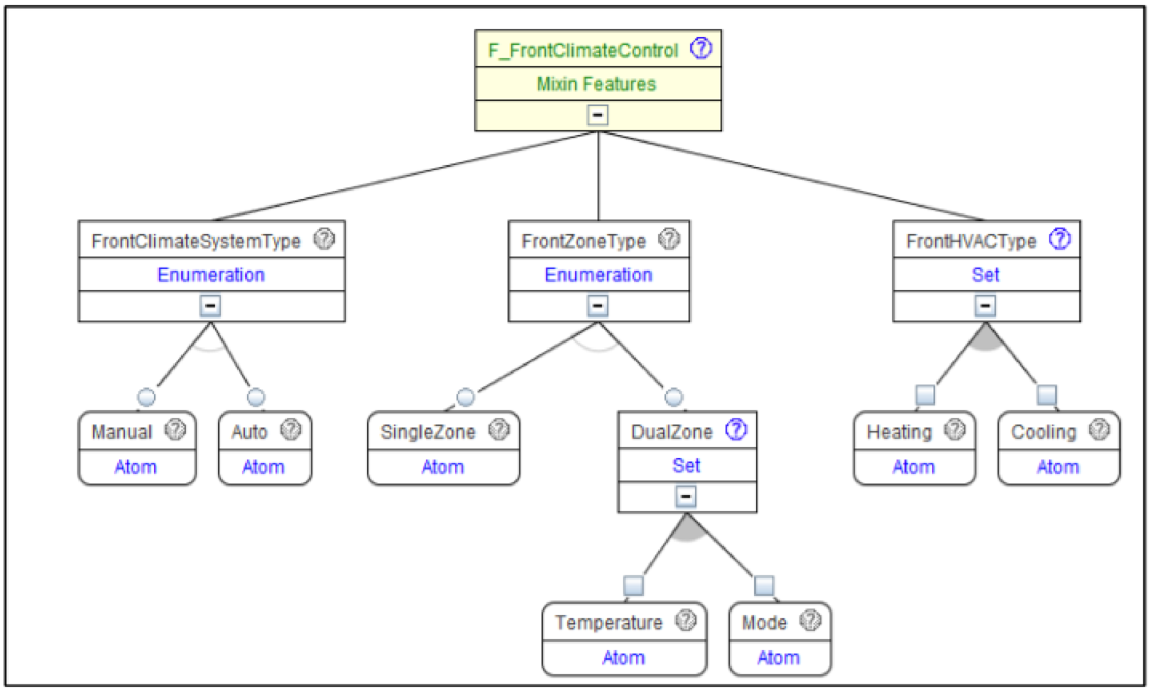


Figure 2. An example of Feature Model

3. Define profiles and make the choices, which means we select the required features. All the user selections from the feature model together are called as profile and, in this case, it is

ManualSingleZoneHeatCool. For this profile, FrontClimnateSystemType is set to Manual, FrontZoneType is set to SingleZone and FrontHVACType has both heating and cooling as shown in Figure 3.

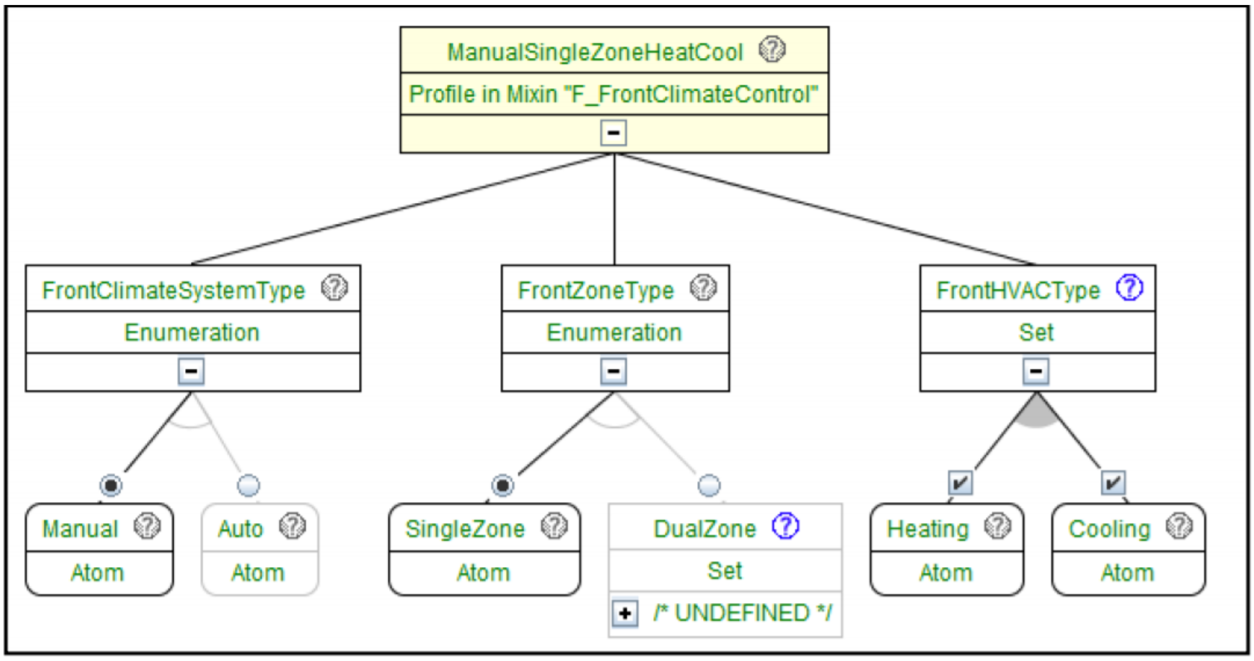
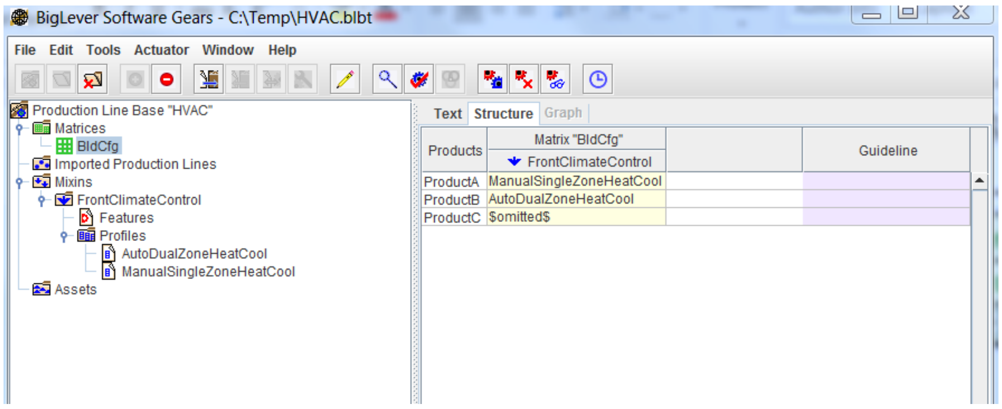


Figure 3. Feature Selection in Profiles

4. Map profiles to Products. As profile selection is done, now the same need to be mapped to a product. As shown in the figure 4, ManualSingleZoneHeatCool profile is mapped to ProductA.



Products	Matrix "BldCfg" ▼ FrontClimateControl		Guideline
ProductA	ManualSingleZoneHeatCool		
ProductB	AutoDualZoneHeatCool		
ProductC	$omitted$		

Figure 4. Matrices where profiles are mapped to products

a. Products are on the first column and the other columns are Mixins/Rhapsody Assets

(Mixin - A mixin is an importable package of feature declarations and feature profiles)

b. Profiles are chosen for Mixin as per the needs

5. Configuring Rhapsody model to add variation points (one-time)

a. Add Gears Profile to Model

b. Connect Rhapsody model to desired Gears model

6. Start adding variation points in Rhapsody model and its logic on different model elements.

a. Select an element in the Rhapsody model and choose “Edit logic”. Edit logic dialog gets opened as shown in Figure 6

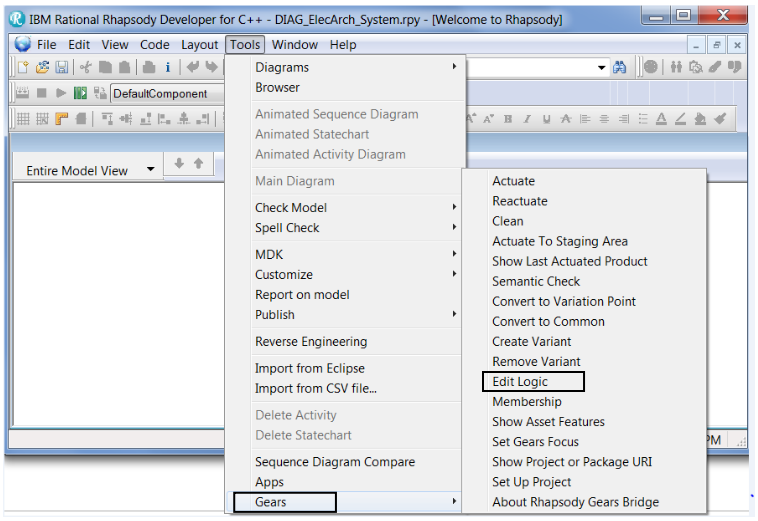


Figure 5. Option to add variation logic

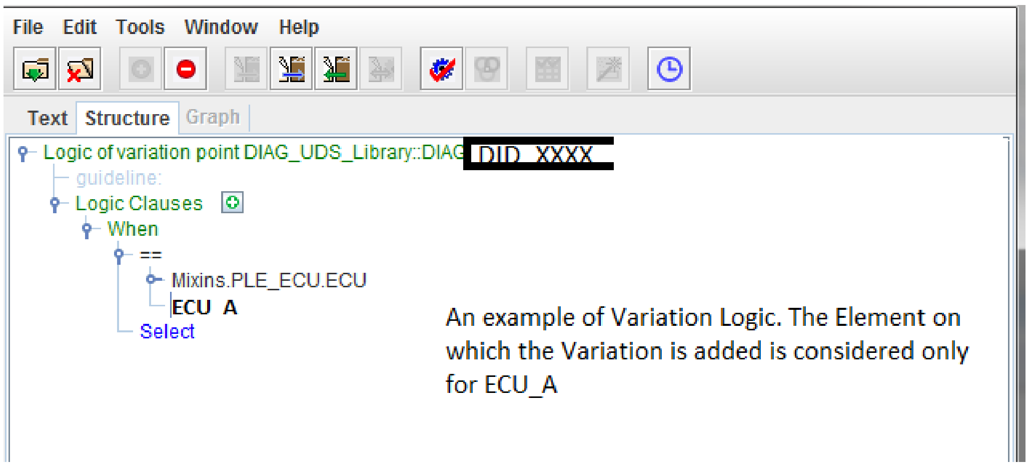
With the above setup, Variation points can be added on any model element and apply Variation logic.

Variation Scenarios:

a. Variation at ECU level: To be generate different ODX files for different suppliers

b. Variation at Object level: A diagnostic object say a DID need to have Read/Write access for certain ECUs and only Read for other ECUs.

As shown in the Figure6, If the generation is for ECU_A, the logic on DID_XXXX will evaluate to true and is considered for generation. For any other ECU, the conditions result false and not considered.



```
Logic of variation point DIAG_UDS_Library::DIAG DID_XXXX
  guideline:
  Logic Clauses
    When
      ==
        Mixins.PLE_ECU.ECU
        ECU_A
      Select
```

An example of Variation Logic. The Element on which the Variation is added is considered only for ECU_A

Figure 6. An example of Variation Logic

7. During the ODX generation, Rhapsody models and Gears models are parsed. When certain element is being processed and if it has variation point, its logic is evaluated and decided to consider it for that ECU or not. So, based on the logic, a signal/DID/RID/DTC might be generated in the ODX or eliminated.

Summary

In this paper, we discussed the issues of defining diagnostic identifiers for complex automotive vehicles and described the solution with variation management along with example. We discuss different tools like Rhapsody, Big Lever Gears and Rhapsody Gears Bridge that we used for deploying variation and the work flow that we follow to define the variation points and generate ODX files for ECUs. This way we reduce man power, bigger models, developmental cost and can improve productivity and quality. We can further use this process to apply variation at object level, ECU level etc. within a system.

References

Anilloy Frank, Eugen Brenner, *'Model-based Variability Management for Complex Embedded Networks'*, 2010 Fifth International Multi-conference on Computing in the Global Information Technology.

Fabian Kliemannel, Georg Rock, Stefan Mann, '*A Custom Approach for Variability Management in Automotive Applications*'.

Charles W. Krueger, *'Variation Management for Software Production Lines'.*

BigLever Software Gears User's Guide.

Abbreviations

ECU	Electronic Control Unit
DID	Data Identifier
RID	Routine Identifier
DTC	Diagnostic Trouble Code
UML	Unified Modeling Language
XML	eXtensible Markup Language
SysML	Systems Modeling Language
PLE	Product Line Engineering
ISO	International Organization for Standardization
AUTOSAR	Automotive Open System Architecture

Biography

Aswin Nata. He is working as a Lead Engineer in General Motors Technical Centre India Pvt. Ltd, Bangalore. He holds post-graduation degree in Software Systems and have been working in General Moros for 11 years working in Tools development used in work cycle of Software development. Currently working in in GM Architecture group as lead developer in Model based Systems/Software development that includes ODX Generation.

Eswar Kumar Kodati. He is working as a lead Engineer in General Motors Technical Centre India Pvt. Ltd, Bangalore. He holds post-graduation degree in Electronic Instrumentation Engineering and have been working with General Motors for 11 years. Eswar has technical expertise in Automotive communications protocols in CAN and LIN. He has extensive experience in developing infrastructure software for ECUs and on vehicle diagnostics. Currently, he is leading diagnostic team under GM Global Electrical Architecture Group to deliver ODX files for all GM vehicle ECUs.