Modularity And Associated Tools As A Mean To Master Quality Of Complex Embedded Systems

Robert Bonetto, Jean Michon

Modularity And Associated Tools As A Mean To Master Quality Of Complex Embedded Systems

Robert Bonetto[1], Jean Claude Michon[2],

1: RENAULT
2: FCF

Abstract

In line with its focus on quality, RENAULT has settled a special action plan to answer the burst of complexity of engine management systems : EMS 2010. It will allow us a very high mastery of complex systems' development with limited efforts in term of development time or development costs. This will enabled us to concentrate our forces on the benefits for our customers. Its development has been supported by our main engine management systems' suppliers and especially the CONTINENTAL company.

The basic feature of EMS 2010 is a modular architecture. This architecture is answering functional and dysfunctional requirements, and takes into accounts the constraints of embedded real time applications. The main objective is to answer all our vehicle line-up technical definitions with a limited number of versions of standardized modules. These modules have a standardized interface and adapt themselves to several parameters thanks to a particular mechanism. The second objective is to have standard modules of code that can be re-used on all our engine management electronic control units (ECU). The modules are hardware independent, thanks to specific coding rules, and are "plugged" on the basic software of ECU.

The key element is our shelf that records our modules, and all the data linked to their development and their validation, allowing a high level of mastery in their development. The basis of the shelf, is a configuration management tool ; it manages the different versions of the modules but also their automatic adaptation to vehicle and engine technical definition. It supports also the development processes of our modules and is structured according to our architecture.

Some specific tools are linked to this configuration management tool, so as to master the important features or development processes of our modular product :

- Architecture : data flows verification , real time operation, specification rules
- Dysfunctional operation : failure propagation, dysfunctional process management
- Specifications and models : model in the loop validation, specifications rules checker
- Code : software in the loop validation, coding rules checker
- Calibrations
- Issues management, change management, project management…

---

1. Introduction

The car market is experiencing a very strong competition and a significant increase in customers' expectations. A first consequence has been a very important increase of the performances of internal combustion engines, inducing a very quick development of the capacity of the engine electronic management. The engine is operated on very precise conditions, to optimize its efficiency, increase its reliability, ease its reparability and reduce its emissions at the same time. It requires the computation of a lot of information, coming from many sensors, and a very significant number of adjustable parameters, requiring many actuators. A second consequence is the multiplication of models and versions of cars, sold in different countries, with different regulations. This has also an impact on engine management systems, which have to comply with many different requirements. Thus, the complexity of the embedded software has been multiplied about tenfold during the last ten years.

As far as RENAULT is concerned, the mastery of electronic management system has historically been a point of high interest. In line with the company

focus on quality, a major objective of RENAULT contract 2009, RENAULT has settled a special action plan to answer the burst of complexity of engine management systems : EMS 2010. Its development has been supported by RENAULT main engine management systems' suppliers and especially the CONTINENTAL company, that has been our main partner for this project. BOSCH and VALEO have also largely contributed.
First objective of the plan is the mastery of quality, despite the increase of complexity. Second objective is to reduce the development cost for a given level of complexity. We will see later how we have been able to achieve at the same time this better quality with limited development costs.

This plan had two main axis :

- Mastery of complexity thanks to a structured architecture. This architecture should allow confinement in order to identify and limit the impacts of changes. It also should increase the standardization of the different elements of the architecture
- Implementation of rigorous processes, adapted to an industrial production.

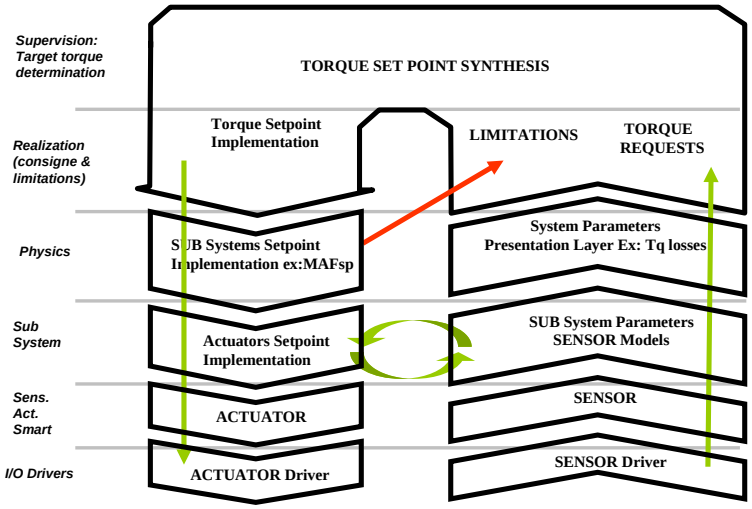
We will now focus on the first one, the architecture.

2. Functional architecture

RENAULT, as a major engine manufacturer, designs with a combined process its engines and the way the electronics has to operate them : this allows the best optimization of the complete powertrain. The strategies to operate the engine are the core of what we call the Engine Management System (EMS).
In order to master these strategies it is important to give them a clear, logical and strong structure.
The first principle that has driven the design of our architecture is that the functions should be spread in different layers, with a progressive abstraction of the hardware.

- first layer makes the abstraction of the different sensors or actuators of the system enabling the upper functions to be little sensitive to changes of type of sensors or actuators.
- second layer deals with subsystem operation such the realization of a given Exhaust Gas Recirculation valve section or a given throttle opening.
- third layer deals more with the physics of the engine. It will realize the operating conditions of the engine : decide the EGR rate, the air flow.
- The upper layer decides the torque that the engine as to deliver, taking into account all the needs (driver, losses, air conditioning, transmission, stability control…).

From this structure it is clear that elements are allowed to exchange information if they are in two consecutive layers or on the same layer. The other flows are generally forbidden.



This structure has enabled us to limit the dataflow between modules thus implementing a strong confinement between the elements of the architecture.
The next question was, how to implement this functional architecture in our EMS.

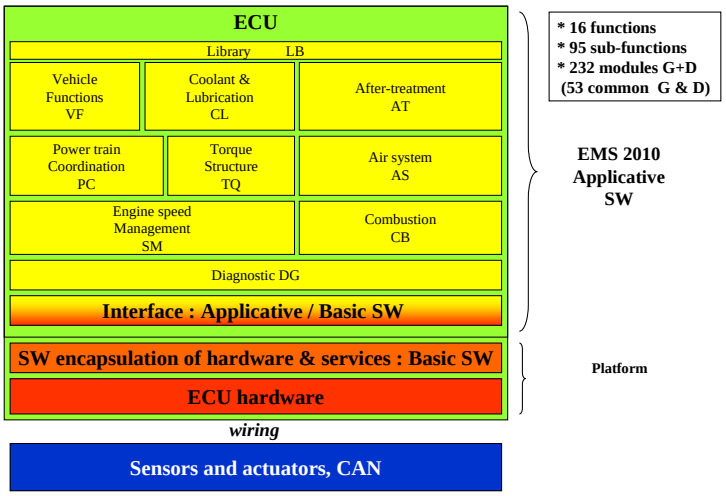
An EMS is first a set of electro-mechanic and electronic elements (sensors, actuators, wiring and the Engine Control Unit) that we procure from suppliers, according to our requirements.
Then, in the Engine Control Unit (ECU) runs a software (SW). In this SW we can distinguish according to our point of view two main parts.

- basic SW: it implements the services necessary to run the applicative part and to operate hardware resources (operating system, communication services and sensors and actuators' drivers)
- applicative SW, running the strategies that pilot the engine. We have seen that we have structured these strategies according to several layers. This structure has been enforced in the applicative SW through modules.

The module is the smaller element of the architecture. It has its own functionality.

In the applicative software the modules are grouped in 95 sub-functions and 16 functions, according to their functionality.
We have also developed an interface between our applicative standard software and the basic SW we buy from each supplier. The main purpose of this interface is to « adapt the impedances » between our standards and the supplier's one.

3. Development process



How to develop an engine management system, taking advantage of our architecture ?

- First we have produced a shelf of modules with the structure of our architecture. In each compartment we find one module of specifications (Matlab Simulink models), the associated C-code, the elements allowing the use of the module on a platform and the important design documents such as the validation reports. All the elements are of course under configuration management.
- From the project specifications we identify the modules we have to create or to modify, according to our architecture, taking as a constraint to minimize the impacts on existing modules
- The models are designed and validated through several means, including simulation (Model In the Loop) or rapid prototyping. We have implemented specific modular design rules, that are checked with a specific tool.
- The models are coded by suppliers. We check the compliance of the code with our modular coding rules in order to guaranty the code independence to the hardware and the respect of our architecture. Then the functionality of the code is validated. For the modules that benefit of a MIL validation we automatically check that the code behavior is the same as the model's one with a Software in the Loop tool.
- The validated modules are placed on the shelf. During all this process a specific tool, EMSET, guides our teams through the different steps, ensuring its repeatability and quality.
- The architects define the set of modules for each project. The EMSET tool configures the modules according to the project technical definition.
- On the same time ECU suppliers have developed the necessary platforms (hardware, basic software and interface) for the projects.
- The set of C-code modules (applicative code) and relevant documents are sent to these ECU suppliers. They integrate them on their platform and valid their correct integration.
- We validate the platform and the applicative code and then the calibration process starts on the vehicle

The basis of this development process is the re-use of validated modules of C-code enabled by the structuring of our applicative SW. The advantages are easy to understand :

- Re-use of proven-technology with cumulative validation : quality
- New projects development limited to new functionalities whatever the ECU supplier : cost and development time

4. Upstream V

As already mentioned, the benefits gained from a standard architecture are considerable. But this standard architecture cannot be an obstacle to innovation : it has to evolve with a large spectrum of possible modifications from minor rectification (you cannot expect architecture to be perfect) to integration of big new functions (particulate filter has been such a big new function, and NOx after treatment is today).

For new innovative functions, a specific process has been defined (Upstream V). At the very beginning of this process is an analysis of the impact of the introduction of the new function. Thus the modified architecture is defined in order to minimize the impacts in terms of modifications on functional and dysfunctional data flow, number of impacted modules, and complexity of modifications. The result of this analysis is an optimized (from architecture

point of view) set of modules, defined by their external interfaces, and empty at this stage.

The key technical object for this operation is the Module Interface Definition (MID) containing for each module :

- Information linked to technical diversity management
- Inputs, Outputs (functional flow)
- Dysfunctional interfaces and local fail-safe modes
- Scheduling
- Other features under investigation, such as Functional FMEA (Failure Modes and Effects Analysis) for the module.

This technical process has to be managed. A single centralized entity, composed of functional architects has the ability to make decisions about architecture changes. The main responsibilities of this team, as keepers of standard architecture, are:

- Evaluation of different solutions for new functions and final choice
- Decision making regarding minor architecture changes requested by the projects
- Management of specification rules (as part of the architecture) and the associated verification tool.

Due to their skills and knowledge of architecture, this team is also involved in different tasks for industrial production of software such as global dataflow control, or real time global architecture building. These tasks are described later in this paper.

This way to manage introduction of new functions enables also different business models for the development of these functions:

- In house development
- Delocalized development in another RSA technical center
- Subcontracting
- Purchase of already developed function

In any case, the early definition of modules and associated MIDs gives a very high level of confidence for plugging (integration) of the new function in the architecture.

Another very important feature of EMS software is the reconfiguration management (shift to degraded or fail-safe modes) in order to guarantee the mastery of safety and availability.
The management of failures is based upon a dysfunctional architecture, consistent with the modular functional architecture.
The principle is simple: each important functional dataflow has an associated Validity Level Indicator (VLI). Each module has to react according to the validity of its inputs, and to propagate the failure indication by the setting of appropriate VLI for its outputs.
The input modules, close to hardware and OS generate VLI according to electrical and functional diagnosis. Intermediate modules generate VLI according to their VLI inputs, and final setpoint elaboration modules choose the best degraded or fail-safe mode.
A global evaluation of this dysfunctional architecture is made, using the appropriate tools to :

- Check the global consistency of local and global degraded or fail-safe modes
- Identify lacks in the specifications, for diagnosis and compliant treatment or generation of VLI.

Of course, all these mechanisms are based upon :

- FMEA for each component of the system, identifying the possible origin of failures
- A global system FMEA to assess the coverage of failures against system objectives, for the whole system.

5. Software industrial production

Software industrial production is based on a tool managing the shelf : input in the shelf for new modules, and output for EMS software production.

In order to understand the extent of complexity for shelf management, we have first to comment the origin of technical diversity. The main axes for diversity are:

- Major Technical Definition features: Gasoline, Diesel, NoxTrap, … These big options are managed through branches of architecture. Nevertheless, the architecture has been designed in order to maximize commonalities for example between Gasoline and Diesel, and a significant number of modules are common (see example later).
- Vehicle architecture. For example, different families of vehicles have different on CAN messaging system. Again the architecture has been designed in order to minimize the impact of these differences.
- Minor Technical Definition features. The main examples are: use of a sensor or use of a model for some parameters, different types of sensors for the same physical parameter (Threshold measurement or continuous measurement). The mechanism used is option integrated in a module. For

the designer of a software system, the available choices are:
    - Select an option at design time. In this case, only the selected part of the module will be embedded in the final product.
    - Delay the selection and embed all options. In this case, the code for all options is embedded in the final product, and the applicable option is chosen by a configuration calibration.
- Legal aspects: OBD is the best example. Options are depending on local regulations.
- Hardware platform. The code of the majority of modules is portable, and identical for different hardware platform. But the interface with OS and vendor specific low level functions is not portable.
- Precise matching and tuning for a vehicle: calibrations (for configuration of modules and for adaptation to engine thermodynamic).
- And of course module version management.

The tool used for shelf management is named EMSET for Engine Management System Engineering Tool. Basically, it is a standard configuration management tool, customized for the specific framework of EMS2010 project. It is linked to task dedicated business tools, for module input and software output.
The basic object in EMSET is the module with :

- Specifications
- Module Interface Definition (MID)
- Code
- Models for specification and code validation
- Results of validation, …
- All these elements are versioned.

EMSET integrates of course the classical services functions to manage the usual life of software objects : issues management, change management, project management. But the main goal of EMSET is to provide the two main functions described below.

The first main function of EMSET is dedicated to module development with :

- Check of specification rules, by a dedicated tool (These rules are part of architecture, and their configuration is also managed by EMSET)
- Specification validation, based upon models, by a dedicated tool (MIL)
- Check of coding rules, again by a peripheral dedicated tool. Coding rules are very important to insure portability of modules on different hardware platforms.
- Code validation, using the model already developed for specifications validation
- Check of VLI use and production.

The second main function of EMSET is dedicated to software design (integration of on the shelf modules), with :

- Ability to choose the modules composing the product
- Ability to choose for a module:
    - The version, according to technical definition, level of functionalities, validation level, …
    - The options for technical definition options: embedded or not …
- Support for consistence analysis for the chosen modules.

This last item deserves a larger comment. The number of possible combinations of modules, including configuration options and versions does not enable a comprehensive management of all these combinations. Our choice is:

- Beforehand, a partial preliminary management of consistencies between modules (versions included) limiting the possible choices.
- Consistency checks of the set of modules composing the software product, afterwards. The main checks are:
    - Functional dataflow consistency (no input or output pending, …)
    - Dysfunctional management consistency (propagation of failure messages)
    - Real time management consistency (usage of events or recurrences)

In order to illustrate the nature of these checks, we will describe a tool used for Dysfunctional Consistency Checking.
Each module producing an output with VLI specifies the different possible values and their meaning.
Each module using an input with VLI requires the different possible values it needs (with their meaning).
The consistency between a producer of information and all consumers is recorded in a LNA (Level Needs Adequation). This LNA can change, or not, when the version of a module changes.
So, a simple analysis of LNA enable to check the consistency between two versions of modules, from a dysfunctional point of view.
This tool is used:

- For the choice of a module version at design time
- For a global check of the product, after the selection of all modules

6. Conclusion

The EMS 2010 project is now in its final phase. At least one version of each module is present in our shelf. This enables us to build the applicative software for the first projects that will be released soon.
Let us take now an example. One of these projects will be a diesel engine equipped with a particulate filter to fulfill the next emission regulations with a very good fuel efficiency.
The applicative SW of this project contains 150 modules. These modules have been coded by the three partners of the project, CONTINENTAL, BOSCH and VALEO.

We can distinguish 4 categories of modules

- Interface modules : these modules are generally very simple also rather numerous. There could be a way to withdraw the need of this interface : standardization of interfaces for powertrain sensors and actuators. We are currently working in Autosar 10.2 group in this objective.
- Specific modules, linked to the specificity of application
- Common modules for all RENAULT diesel engines
- Common modules for all RENAULT gasoline and diesel engines

A fifth category does not appear on this scheme : the modules that are common for all our gasoline engines.

Today, these modules are used by more than ten main projects running in parallel. For each module, the lessons learned resulting from the number of configurations, number of test scenarii, accumulated kilometers on vehicles are greatly increased compared to traditional scheme of development, giving us a further enhancement in the mastery of quality that is a constant focus for the company.
The same modules are running on 4 different ECUs coming from 3 different suppliers and using 3 different microcontrollers. This is a proof of the validity of our concepts, but also this will give to the workshops a standard behavior of all of our range of engines, increasing the efficiency of the maintenance and easing the reparability, in line with our concern of giving our customers the best available cost of ownership.
The re-use process reduces also our development effort and duration. It allows us to concentrate our efforts on the added value for our customers of engine management system’s strategies, such as the introduction of new solutions to limit the environmental impact of our vehicles.