Increasing intersystem functionalities validations efficiency thanks to Model-Based Design

Pedro Moreno Lahore, Yves Touzeau, Olivier Guetta

Increasing intersystem functionalities validations efficiency thanks to Model-Based Design

Authors:
Pedro Moreno Lahore, Yves Touzeau & Olivier Guetta (RENAULT).

Introduction/context:

Since the late nineties, the number of ECUs (Electronic Control Units) used in our vehicles has been growing steadily to reach about from thirty up to sixty ECUs for a premium vehicle.

The methodologies used in the development of the ECUs are far from being homogeneous in an automotive company due to the following non-exhaustive list of causes (More or less related between them):

- Distance in organizational charts (Ex: Vehicle and Power-train departments)
- Physical distance (Ex: Location in different technical centers often not in the same cities/countries)
- Controlled Systems technology and functional needs differences
- Technical background & skills of development teams
- Human & economic resources dedicated to the systems' development

Nowadays we can observe that the methodologies used for the ECUs development have strongly diverged.

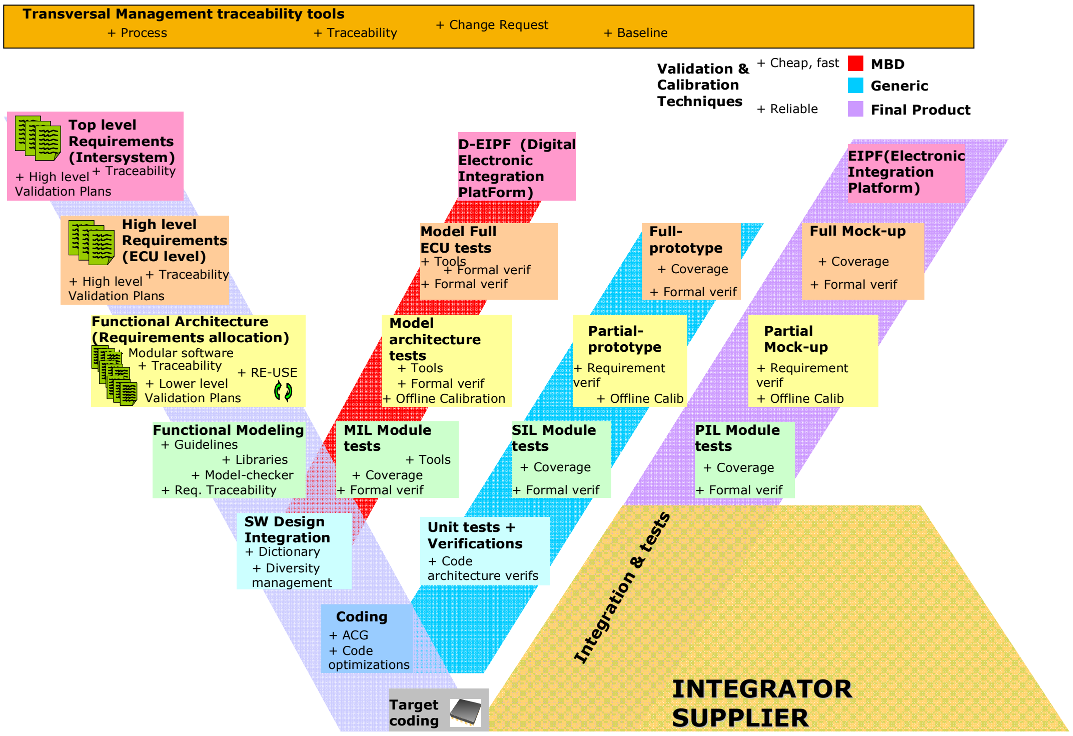
Some examples, little isolated teams use high-level text specifications for ECU development as other teams use Model-Based Design development process and tool-chains. Some teams use the reduced distance between them to create together accurate MBD inter-system tools used for high-level requirements validations. As a result, nearly all the teams are using or developing their own processes, rules and tools for ECU development. Therefore, they often have to correct errors made by other teams in their own company.

ECUs functionalities become more inter-related and complex. The schema in which each developer uses its own rules, process and tools is no longer valid: these functionalities have to be validated before the physical ECUs are available in order to reduce time to market and costs.

Theory 1: Analysis of the tasks used in the V development cycle

In order to understand the difference between the development technologies, we will present a global analysis of the different tasks, methodologies and tools that the Software designers can use along the development process. We will also explain the limitations and advantages of the different tasks.

In this figure, we can see a variant of the classic V development process for the ascending phase.



During the V descent phase, we can identify several phases related to later validation steps:

- Top level Requirements: Definition of the requirements for each of the components which are part of the system. Those system requirements need to be validated with predictive or final means predictive of the whole vehicle high potential electronic via the use of power test benches

- High level Requirements: Exhaustive definition of each ECU requirements. They need to be validated and accepted both in format and content. These requirement validations are performed with predictive or final means of the ECU high potential electronic environment

- Functional architecture: In this step it is necessary to distinguish the requirements allocated to the ECU platform (Hardware + Basis software) and the applicative software. The applicative software requirements must be allocated within functionalities using system architecture criteria. A detailed development phase will allow the definition of the final requirements allocated to each SWC (Software Components), inputs/outputs and each SWC behavior constrains.

- Functional modeling: Means the design of each SWC. The models produced will have to validate both design rules and functional validation of the requirements.

Changes to the classic development process occur in the Ascending part of V cycle phase, as three possibilities are available for the developers:

<u>1st Ascending part of V cycle possibility: Use of Virtual models verification:</u>
This technique is the base of Model Based Design, its main advantages are:

- Anticipation, as this verification can be done soon in the process before costly phases
- Response time, implementing changes and checking their functional behavior can be done within minutes
- Cost, only a computer with an algorithm design and simulation tool is necessary.

First limit to using this technique is the similarity between the algorithm model and the algorithm real behavior in the ECU. In order to improve this similarity, the definition, application and verification of design guidelines are necessary. The use of Automatic Code Generation, in which few or no interpretation of the algorithm model is performed manually by the coders, is very useful for this technique.

Second limit is the similarity and predictivity of the controlled system model providing feedback of the ECU's actions, used to close the loop. Difficulties using such controlled systems models are:

-Low knowledge of the controlled system behavior (Typically in innovation project)

- Controlled system complexity

- Controlled system model predictivity level lower than required for offline development.

In the indicated previous cases this MBD methodology can not be used alone and the next technique is necessary.

<u>2nd Ascending part of V cycle possibility: Use of algorithm generic code to control the real system (Prototype)</u>

The main advantage of this technique is that no model of the controlled system is necessary as the developer can use the generic code to control the real system. In order to use this technique the developer compiles the model in a quick prototyping target.

The first limit is the fact that we have no insurance that the code behavior is equal to our algorithm model behavior. In order to check it, a MIL-SIL (Model/Software in the loop) validation is necessary.

The second limit is the fact that ECUs resources constraints are not taken into account, as the compilation runs in a PC far more powerful than the vehicle ECUs.

The third limit is the cost of the associated vehicle tests used.

<u>3rd Ascending part of V cycle possibility: Use of the final integrated software in its compilation target</u>

This case consist in the use of the final physical ECU (Final product) for the validation. It provide us a proper view of the ECU limitations via the use of PIL (Processor in the loop) and of the high potential electronic via the use of power test benches.

Limitations of using this methodology alone without use of 1st & 2nd methodology are the following:

- Bugs are found very late in the development process, leading to huge expenses
- The software is already integrated and manufactured. As a consequence the corrections of bugs at this step will be expensive and can delay ECU time to market.

In the next chapter we will provide a return of experience for MBD development at a part of the functional level

Return of experience 1: Shared simulation environment for Particulate Filter MBD validation

1. Development Context

The sample concerns the development of the Diesel Particulate Filter control strategies, now integrated on all the Diesel vehicles since the Euro 5 standard application; The main requirement of this technology is to limit the pollutants emissions with the minimum effect on car performances and the minimum impact on the client consumption.

However, the development of this system is very complex, different actors of the system development need to be strongly linked together:

- Requirements definition,
- Strategies design,
- Tuning.

Requirement definition needs to take into account the variety of potential client behaviors to verify that the system fulfills its objectives (particularly security and over consumption linked to the PFT).

However, the potential client behaviors are very different (highway drivers versus city drivers for example). As a consequence, some validation tests can be very expensive, very long or even impossible! For example, in order to estimate the mean over-consumption linked to the DPF for the average clients (and not only on the NMVEG cycle), it should be necessary to take numerous vehicles that are representative of the commercial vehicle and make them realize cycles that are representative of the clients who buy this vehicle over the year (As climate changes have a huge impact on the system performances): in fact, this kind of test is impossible because it would be too long and expensive.
In this case, simulation is not only a way to reduce tests, but the only way to evaluate parameters influence in the requirement target

2. Shared environment Application for MBD validation

In order to response to these development complexities, Renault Power-train department developed a simulation environment tool which is shared by all the entities working on after-treatment systems (strategy conception, calibration definition but also after-treatment requirement definition and validation team).

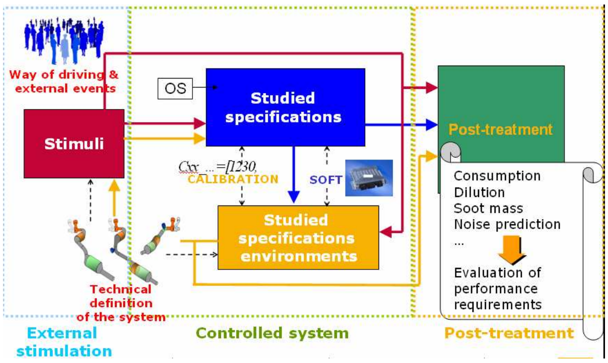
The objective of this simulation platform is to have a common simulation environment which allows us to:

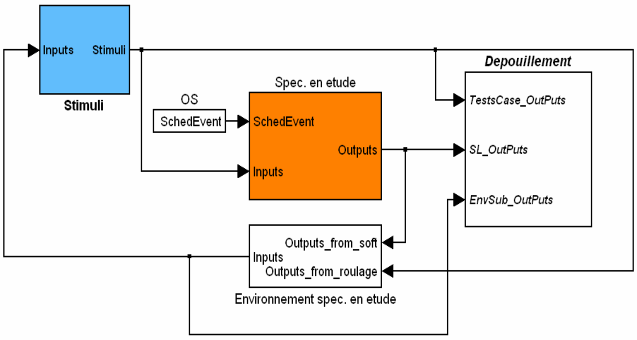
- Define the exhaust line technical definition (PFT volume, material, positions, …),
- Develop and validate the control strategies,
- Tune these strategies,

These calculations have to be made:

- For a representative amount of clients (approximately 200 client) ~= 6 millions kilometers, in different cities and times of the year.
- In a short time ~= 2h00 min

The development/simulation & validation environment has a fixed structure (Independently of the functionality tested) composed by 5 main items.





- Stimuli: External & independent of the studied control system signals. It is part of the requirement definition job to indicate in which conditions the functionality must be tested. In after treatment we use a client statistic behavior database, which allows us to be representative of the way all our clients drive.

- Studied specifications: Model of the algorithms developed to be used in the ECM. Algorithm design is responsibility of the designers while calibrations are defined by the tuning team
- Studied specifications environment: Model of the controlled system which allows us to predict the impact of the software actions and the client behavior on the modeled systems (PFT, injectors, etc). Share of responsibility for this block is complex: Hypothesis accepted for controlled system model should be defined by requirement team. Model is made or bought by algorithm design team or a modification team close to it. "Check: lessons learnt, controlled system model handling"
- Post-treatment: Result box, acceptance criteria defined by requirement team is checked in this box

The use of a shared simulation environment improves cooperation as each team is responsible of a well defined part of model which can be used as an output of their work.

2. Lessons learnt

SILVER (Shared environment tool name) started as an innovation project and is now applied with some difficulties in after treatment systems developments since 2009. The analyses of these difficulties are the key for the successful application of these MBD validation techniques to all electronic systems development.

1st: Differences between the control algorithm and the ECU behavior + difficulties for simulating different control algorithm (~Software Components) together.

Fact: In 2008 when the experience was performed, our design rules did not take into account:

a) Initializations and NVM (Non Volatile Memory) values handling properly. Simulation start was assumed as vehicle key-on and simulation end as key-off. In order to simulate thousands of kilometers we needed to take into account the key-on-off-on events, which meant that all the strategies used needed to be modified (Different from the ones sent to the coder)

b) Ensuring that two different SWC (Software components) could be simulated together.

c) Possible differences between algorithms and the final software appearing typically at the coding step

Solution: This first difficulty is solved by the definition and application of shared strict design rules which takes into account the functionalities that need to be simulated. Using rules which ensure the compatibility with Automatic Code Generation reduces the differences between the algorithms and final software.

2nd: Offline simulation time

Fact: Initially we thought it will be the main issue, simulations would take too long

Solution: Thanks to some design modifications, we made our strategies compatible with compilation; we were able to build a quick simulation executable which solved the issue. The use of design rules compatible with ACG definitely solves this issue. The increase of computer resources available seems to grow quicker than the software size allowing us to think that it should not become an issue in the future..

3rd: Difficulties for re-using the simulation environments, mainly due to management of the controlled system models

Fact: The controlled system model is normally developed during the innovation phases in which we increase our knowledge of the controlled system behavior. Its parameters are obtained via some targeted tests for one specific application. The problem comes when we want to reuse the simulation environment for another application. These controlled systems models need to be parameterized for the new application and sometimes the models adapted

Solution: Controlled systems models need to be managed in version with a proper parameterization methodology as the one used for the algorithm in the ECU. A huge organizational effort is necessary to achieve this goal

4rd: Data exchanges

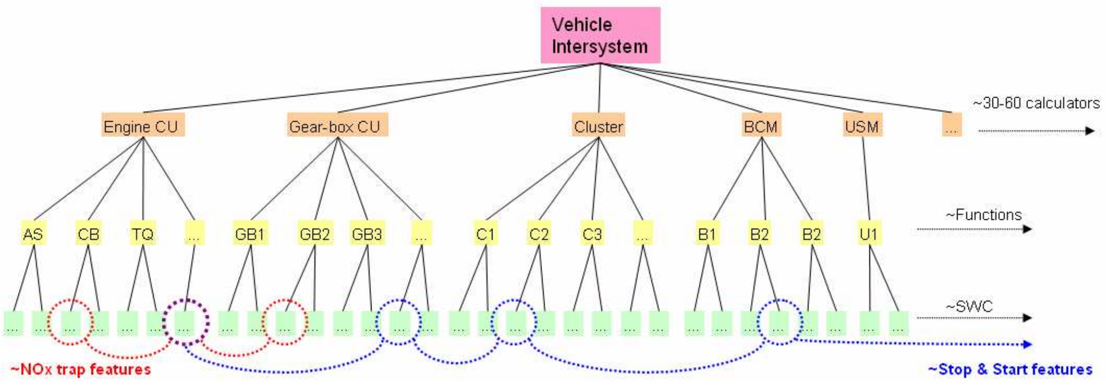
Fact: Using the right software calibrations and variables attributes is difficult. Many issues appeared because in the simulation environment the right calibrations or data where not used.

Solution: Creation of a global database dictionary for ECM in which the information are unique.

Theory 2: Anticipation of intersystem functionalities validation thanks to MBD

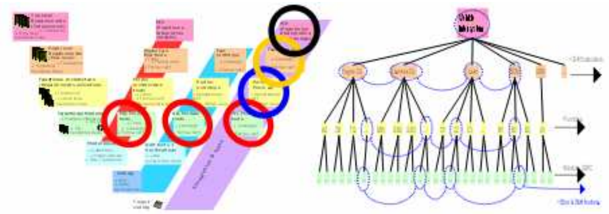
As said in the introduction, the vehicle intersystem functionalities allocated in the different ECUs become more complex with time.

The first “front view” of the V development cycle can be completed with a second “side view” (see next picture) showing the different objects (From ECUs to Software Components) used in the vehicle intersystem development. Historically each of this ECU followed its own V adapted development cycle.



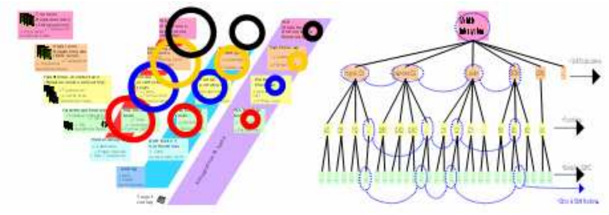
In the figure, we can see two hypothetical functionalities that are allocated along different ECUs.

In order to validate those functionalities the EIPF (Electronic Integration PlatForm) is historically used, in the platform we find all the physical ECUs connected. The EIPF methodology is a possible task in the 3rd Ascending part of V cycle phase of the development process, with all limitations already described for these tasks.



Schema of traditional validation in EIPF, only SWC are validated via MBD; Functionalities crossing SWCs and intersystem functionalities are validated once coding & integration of each ECM is performed.

To reduce the number of validations made with this expensive technique, we would like to apply the same MBD techniques currently used in some single ECUs to the intersystem functionalities validation. This mean of validation is called in Renault the D-EIPF (Digital Electronic Integration PlatForm). The objective of this platform is to detect all functional specification design bugs in order to anticipate algorithm corrections before coding and integration, which leads to costs reduction. The use of D-EIPF should limit EIPF validations to high potential electronic bug validation.



Schema of D-EIPF use. Validations for each granularity level are performed initially at MBD level, reducing the number of validations to be performed with the final object. Require a strong work in requirement definition and validation plans

Using a D-EIPF efficiently is not easy, as we need to ensure that:

1. - All ECUs used in the D-EIPF can be simulated alone, or in a deteriorated form use their code as black box in the simulation
2. - All ECUs can be simulated together
3. – Computers used are powerful enough to design and validate models of that huge size.
4. - ECUs tasks coordination is similar to the one used in the vehicles
5. – We can use software components not developed by our OEM in D-EIPF even if they are black boxes, in worst case develop a simplified model of the ECU

Conditions 1 to 3 means that all ECU algorithm designs must have a degree of maturity close to the one needed for Automatic Code Generation. Most of these ECUs are developed by teams using their own methodologies. In order to ensure ECUs communication a change in their developments methods is necessary.

For those not meeting this level of maturity, changes are necessary and sometimes not accepted by the teams: many of the rules, tools and processes that those teams have been developing for years can no longer be used, and must be common among all the developers involved in order to ensure the efficient use of the D-EIPF platform.

Driving this change is one of the challenges of the Renault Embedded Software expertise domain.

The first steps of this challenge are:

- the analysis of strengths and weaknesses of the different teams
- the identification of blocking points for the deployment of common procedures.

These blocking points may be categorized as technical, economical or human factors. For this last one, convincing such a large panel of developers of changing the way they work is often more difficult than solving technical issues.

In the next chapter we will provide a return of experience of D-EIPF experience

Return of experience 2: The D-EIPF experience.

Description of D-EIPF concept

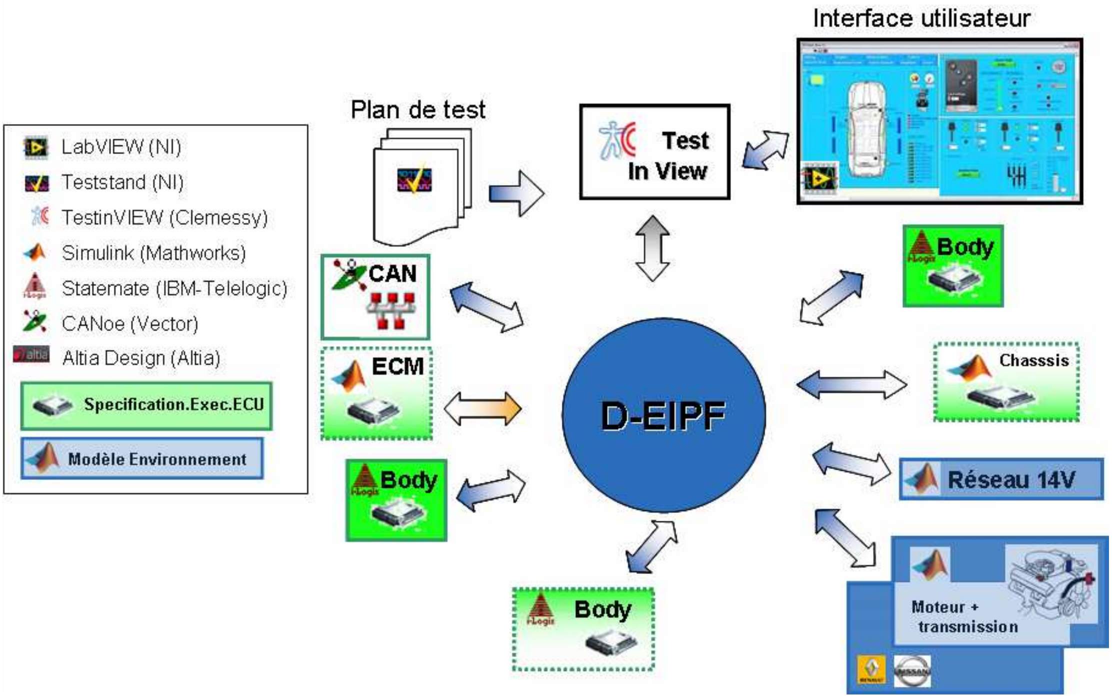
D-EIPF consists on building a simulated system integration platform upstream in the V-Cycle, in order to validate E/E system design and integration of the whole vehicle

In a model-based-design process, the ECU specifications are models. Different types of validation are done with the model:

- Model in the loop validation: Its aim is to validate the specification before sending them to supplier for autocoding. The same test plan is played on MIL than on HIL after reception of ECU from supplier

- D-EIPF validation: The D-EIPF "Digital Electronic Integration PlatForm" is used to validate the integration of ECU in the electric architecture.
  The D-EIPF is especially used for validation of distributed functions (functions that are embedded in several ECU). One again those validation are done before sending models to supplier.

D-EIPF allows synchronous co-simulation. In fact, different models from different languages (Matlab Simulink, Statemate, CANoe…) communicate together and share the same clock.



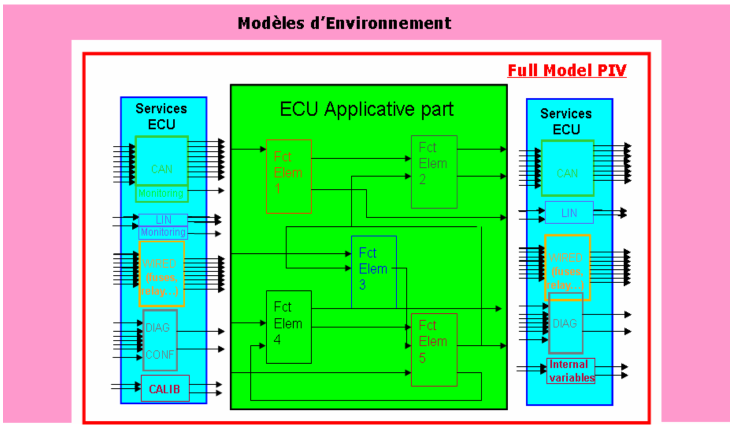
The validation is done by gathering all the runnable specifications of each ECU from the different departments (chassis, body, engine…) and simulating them together including environment models (electrical distribution, vehicle dynamic…). The CAN network is also simulated in order to be predictive in terms of communication.

For the synchronous co-simulation, a specific tool is used: RT-LAB Orchestra from OPAL-RT company.

D-EIPF structure

In D-EIPF there are 3 different kinds of model:

- **ECU full model**: it is the runnable specifications of the Applicative part of the ECU. Some ECU Service parts (simplified models of basic SW and HW parts) are added to the model in order to be able to validate the same way ECU model and real ECU. In fact, the I/O of the model should correspond to the I/O of the ECU.



- **Plant models**: it is model used to simulate the ECU environment. It can be a model of sensor and actuator, or the model of the CAN network communication, or the engine model if it is needed.

- **Part ECU model**: it is a simple representation of the behavior of the ECU but it is not the specification itself. It is used to replace the ECU specification.

D-EIPF main goals

- **Quality improvement**
  - More validation in design phases (short loops)
  - Earlier integration of systems (Inter-system, Naming rules)
  - Fault injection is easily made possible / Proposal to be changed into: Early test of reduced mode

- **Costs reduction**
  - Allowing to detect bugs in Specifications and System Integration far upstream (before sending specifications to supplier) in the V-Cycle and reducing physical validation phases
  - Minimizing risk of remaining bugs.

- **Development Time**
  - Test plan preparation in advance phases
  - Mastery of the systems complexity
  - Fast and economic investigation of various E/E Systems and architectures

Validations done on D-EIPF

It is very important to distinguish between MIL validation and D-EIPF validation.

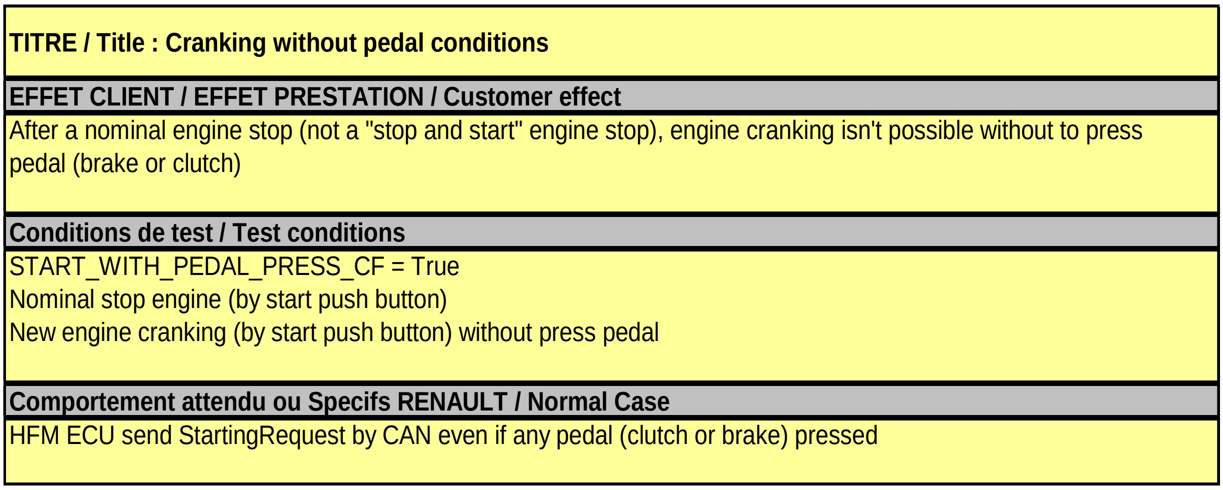
MIL validation consists on the validation of the ECU itself by stimulation of its inputs and outputs. The test plan used should be the same as HIL validation.

D-EIPF validation consists on integration tests. The tests played on D-EIPF are customer-oriented tests. The aim is to validate the right integration of ECU in a vehicle environment, the correct communication between the different ECUs and the requirement compliance of functions distributed on several ECU. The stimuli of the tests are customer actions (door opening, brake pedal pressing…).

Therefore, the anomalies found on D-EIPF are not the same as anomalies found on MIL bench. In fact, the ECU model is validated before its integration on D-EIPF, thus the anomaly found is incompatibility between different ECU specifications.

Example of anomaly found on D-EIPF

This anomaly was not found, as expected, on MIL ECU bench, as the predictions of the intersystem were not precise enough (In fact, on MIL ECU bench, HFM ECU has been validated with the expected behavior BCM ECU, and not the real behavior model BCM).



TITRE / Title : Cranking without pedal conditions
EFFET CLIENT / EFFET PRESTATION / Customer effect
After a nominal engine stop (not a "stop and start" engine stop), engine cranking isn't possible without to press pedal (brake or clutch)
Conditions de test / Test conditions
START_WITH_PEDAL_PRESS_CF = True Nominal stop engine (by start push button) New engine cranking (by start push button) without press pedal
Comportement attendu ou Specifs RENAULT / Normal Case
HFM ECU send StartingRequest by CAN even if any pedal (clutch or brake) pressed

CONCLUSION: Embedded Software Expertise Domain actions to improve intersystem functionality validation.

2 years ago, Renault top management decided to launch again some “Strategic Expertise Area (SEA)” domains. One “Embedded Software Technology” domain was created, showing the importance for an OEM to master software development due to the increasing volume of embedded software in our cars. Experts and specialists from several divisions (mechanical, vehicle, advanced engineering) have been appointed and are now fully or partially dedicated to this transversal and Renault worldwide SEA. One of the main missions consists in defining methods, process and tools to ease the development of software and ensure transversality, consistency and deployment of best practices, which are useful especially for little isolated teams.

Model Based Design is considered as a strategic project for Renault, many people both from expertise and metier are involved. For instance, activities such as definition of a new process for software design, definition of new rules for modeling and common libraries, launch of a common and unique design verification tool, or launch of a global database for requirements and validation plans, have started and are still on-going. The main objectives are reduction of number of bugs found late in the development, and reduction of time needed for the complete design and validation of these embedded software. Furthermore, having common methods and tools used in every division of Renault also eases the transfer of people from one team to another.