SysML for embedded automotive Systems: lessons learned

J-D Piques, E Andrianarison

SysML for embedded automotive Systems: lessons learned

J-D. Piques[1], E. Andrianarison[2]

(1): Valeo - Powertrain Systems Business Group – Electrical Vehicle Product Group
(2): Valeo - Group Electronic Expertise and Development Services
14 avenue des Béguines, F-95892 Cergy-Pontoise Cedex

Abstract: This paper deals with the first lessons learned from using the SysML language to support the System Engineering activities when developing automotive embedded systems and products with a particular focus on illustrating improvement solutions that have been experimented and validated in Valeo pilot projects.

Keywords: Model Based Engineering, System Modeling, SysML, System Engineering, SysCARS

1. Introduction and overview

Motivations

Increasing complexity of technical systems, business models and safety regulation (ISO26262) requires higher formalization effort.

The Model Based System Engineering (MBSE) approach is a key lever for automotive lean processes to cope with this context and still ensuring flexibility and R&D efficiency on innovative products.

Main lessons learned

Although SysML has become the de facto standard for MBSE, a supporting methodological background was and is still mandatory. The SysCARS methodology [1], which is summarized in **Part 2**, defines the sequence of SysML diagrams and artefacts to be released in order to implement the engineering process. However pilot projects have shown this was not sufficient and other critical issues have been addressed.

A major issue is the adoption of SysML existing modelers which are too complicated for non software engineers, providing no guidance on which diagram and artefact to use among overloaded GUIs. To support adoption and deployment control, a workflow driven approach is described in **Part 3** and is implemented by a Valeo profile including ergonomic macros for Artisan Studio modeler.

Moving from a document centric approach to model based engineering shall also ensure formal coupling to requirement related tools. **Part 4** addresses these aspects together with strategy regarding traceability checks and connection to tools such as DOORS and Reqtify.

Still to facilitate adoption and due to weaknesses of SysML compared to discipline modeling / simulation tools, SysCARS support synchronization of structural diagrams. This feature is described in **Part 5** and is used to perform behavioural studies in legacy tools such as Simulink.

Finally **Part 6**, summarizes issues related to system and safety engineering coupling and presents mechanisms supporting "Safety In the Loop" approach (**SaIL**) targeting FMEA/FTA automation.

2. SysCARS methodology overview

SysCARS (System Core Analyses for Robustness and Safety) is a Valeo methodology which provides a practical help for system designers on how to perform the sequence of System modeling activities with SysML. This methodology, detailed in a previous paper [1], is shortly summarized here.

2.1. SysCARS principles

SysCARS methodology added value consists in:

- Selecting a subset of SysML diagrams and artefacts to be used in a convenient and pragmatic way (learning curve optimization)
- Providing defined semantics to ensure diagrams meaning and rules for verifying model consistency
- Defining an obvious diagram sequence which ensures modeling efficiency regarding company processes
- Implementing stereotypes and templates for automatic documentation generation at each stage of the process
- Taking into account coupling constraints with other processes or tools such as Reqtify from IBM for requirement traceability or Simulink from The Mathworks for functional modeling

The current methodology is therefore targeting the optimum trade off for Valeo deployment and is built from existing state of the art. It does not claim for any theoretical novelty, while having merged relevant best practices from existing approaches, such as EIRIS methodology [2]. This implementation is also taking maximum benefits from available features of the selected SysML tool, namely Artisan Studio from Atego.

2.2. SysCARS generic workflow

The overall System Engineering process begins with analyzing the project context, considering the system to be developed as a black box, and then successively goes deeper into the details until specifying internal component features. More precisely the SysCARS methodology is divided into five major phases:

- ***Stakeholder needs definition***
- ***Requirements analysis***
- ***Logical architecture design***
- ***Physical architecture design***
- ***Components needs definition***

For clarity purpose, the process and the sequence of activities are described in a pure sequential way. However, in practice, different steps could be performed simultaneously with iterative and mutual refinements.

Moreover, each phase systematically ends with:

- Traceability analysis, to check the consistency and completeness of activities performed and artefacts created,
- Automatic generation of a document making a synthesis of the activities performed (SND: Stakeholder Needs Document, SyRD: System Requirement Document, SyDD: System Design Document, CND: Component Needs Document).

2.3. SysCARS optimized workflows

The SysCARS workflow is described below.

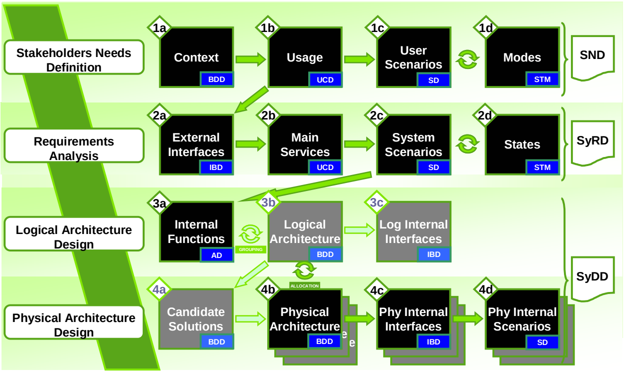

Stakeholders Needs Definition
1a Context BDD
1b Usage UCD
1c User Scenarios SD
1d Modes STM
SND
Requirements Analysis
2a External Interfaces IBD
2b Main Services UCD
2c System Scenarios SD
2d States STM
SyRD
Logical Architecture Design
3a Internal Functions AD
GROUPING
3b Logical Architecture BDD
3c Log Internal Interfaces IBD
ALLOCATION
SyDD
Physical Architecture Design
4a Candidate Solutions BDD
4b Physical Architecture BDD
4c Phy Internal Interfaces IBD
4d Phy Internal Scenarios SD

Figure 01: SysCARS System Engineering Process

The last stage (Component Needs Definition) has not been represented, because it is mainly an extraction of component artefacts from the physical architecture.

The kind of diagram used at each step is given by its SysML acronym attached to the related activity block: Block Definition Diagram (BDD), Internal Block Diagram (IBD), Use Case Diagram (UCD), Sequence Diagram (SD), STate Machine diagram (STM), Activity Diagram (AD)

Lessons learned on pilot projects have shown that in most situations it makes sense to bypass the elaboration of the logical breakdown and to directly allocate internal functions onto the physical architecture blocks. Indeed, physical architectures are very often frozen because resulting from carry over products and therefore the investigation of several candidate solutions is not necessary.

Consequently, two kinds of optimized workflow have been defined depending on the project typology:

- **SysCARS-XS** (eXtended Stream): For innovative products, the whole set of activities of the [figure 01] are performed, and in particular the investigation of several physical architectures.
- **SysCARS-CS** (Core Stream): For carry over products, the activities represented by grey boxes on the [figure 01] are not performed.

3. Workflow-driven approach

3.1. A specific profile for customizing SysML

GUIs of SysML existing tools remain too complicated for a non software specialist, which is the targeted audience for System Engineering. Indeed, SysML user interfaces provide confusing and unneeded features from the UML world. Very often, UML and SysML artefacts and diagrams are mixed without any possibility for the user to limit to a pure SysML scope. Moreover, no guidance is provided on the relevant diagram to be used and on the correct ordering of operations.

To cope with these drawbacks, a specific ergonomic profile (thereafter referred to as “Valeo Profile”) has been developed, introducing the concept of workflow-driven approach. The basic idea behind the workflow-driven approach is to provide the System engineer with a step by step help throughout the SysCARS engineering workflow. Moreover, at each step of the workflow, only relevant features and diagrams are available in a simplified GUI.

The mechanisms of the workflow driven approach are detailed in the chapters below.

3.2. Workflow diagram navigation

When creating a new model with the Valeo profile, this model directly opens a pre-defined “workflow diagram”. The “workflow diagram” is the central element of the Valeo Profile, defining the sequence of modeling activities to be performed in accordance with the SysCARS methodology [1]. In fact, the workflow diagram is simply a statechart diagram, where states and super-states respectively correspond to elementary activities and main stages of the SysCARS methodology. No more than one elementary state can be active at one moment; i.e. only one kind of elementary activity should be performed. On the workflow diagram represented below, the active state is highlighted in blue.

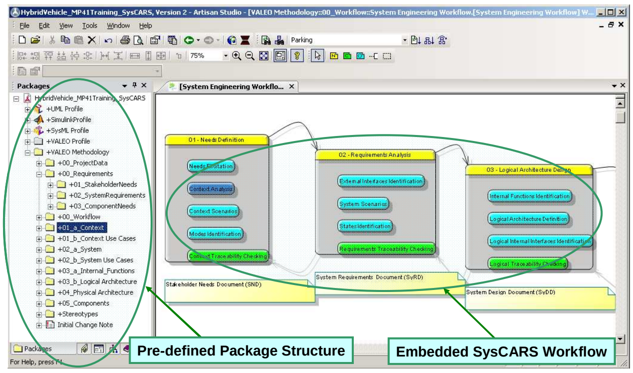

Packages
01 - Needs Definition
02 - Requirements Analysis
03 - Logical Architecture Design
Stakeholder Needs Document (SND)
System Requirements Document (SyRD)
System Design Document (SyDD)
Pre-defined Package Structure
Embedded SysCARS Workflow

Figure 02: Valeo Profile GUI Overview

It is possible to navigate the states of the workflow diagram and to select the workflow commands available: "Next Step", "Previous Step", "Go to step…". Then the modeling step is changed accordingly.

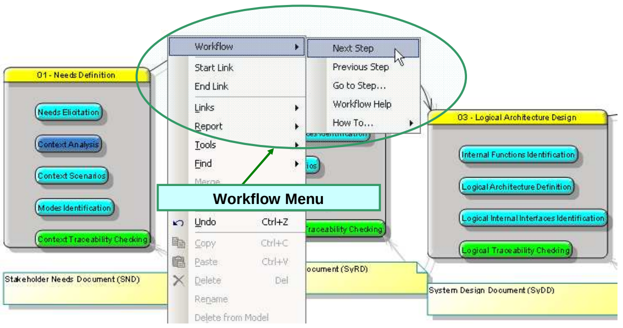


Figure 03: Valeo Profile Navigation

A second kind of navigation mechanism is available from the workflow diagram. Right-clicking on each state allows to reach the diagrams summarizing the results of this modeling step. The relevant diagrams should have been attached as associated diagrams once created.

The implementation of the workflow in the profile is not frozen but configured using a dedicated XML file. This option enables further evolutions on the SysCARS workflow.

3.3. Pre-defined package structure

When creating a new model with the Valeo Profile, this model is also provided with a pre-defined package structure. This package hierarchy is directly correlated to states and super states of the workflow diagram, which in turn correspond to stages and steps of the SysCARS methodology.

However, the user is free to organize differently artefacts and diagrams within a different package structure.

As previously, the pre-defined package structure is not frozen but configured using a dedicated XML file.

3.4. GUI features defined by workflow state

The current active state of the workflow diagram is used to monitor the look and feel of the SysML modeler tool, in order to provide the user only with the features required at this step of the system modeling process. Consequently, command menus available in the object browser and toolbar menus on diagrams are both customized differently in each state of the workflow diagram.

The diagram below clearly shows the level of simplification on command menus reached by the Valeo Profile.

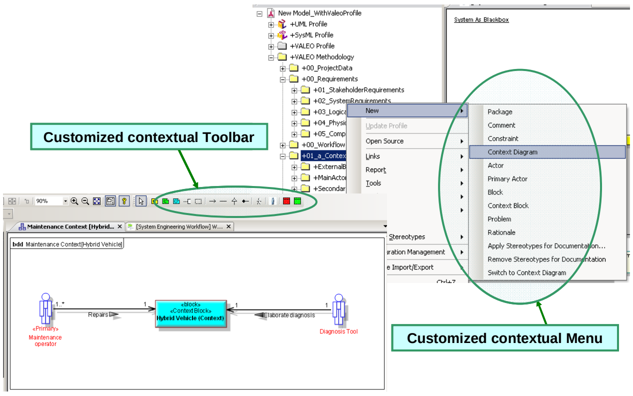

New Model_WithValeoProfile
Customized contextual Toolbar
New
Package
Comment
Constraint
Context Diagram
Actor
Primary Actor
Block
Context Block
Problem
Rationale
Apply Stereotypes for Documentation...
Remove Stereotypes for Documentation
Switch to Context Diagram
Customized contextual Menu
Maintenance Context [Hybrid Vehicle]
Repairs
Elaborate diagnosis
Hybrid Vehicle (Context)
Maintenance operator
Diagnosis Tool

Figure 04: Customized Menus

In the object browser window, the "create" command menu displayed when right-clicking an existing SysML object, is customized individually for each type of SysML artefact and diagram.

In the graphical window, buttons available on each kind of diagram toolbars are also customized depending on the workflow diagram active state.

The GUI features are evolutionary and configured from two dedicated XML files, one for the package browser command menus and one for the diagram toolbars.

3.5. Stereotypes for documentation

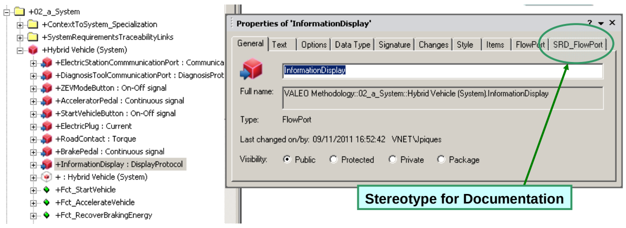
Documentation in a format that is easily comprehensible by a broad range of stakeholders remains an effective way to validate and communicate system design information. The first thing to do is to precisely define the expected document format and contents by creating a corresponding template for the publishing tool. The same document template will be re-used on different projects, without any modification. Then, thanks to the publishing feature of the SysML tool, automatic document generation can be run on demand to collect and format data from the SysML model, without any special effort.

Furthermore, separation between modeling data and document templates enables versatile customisation either to generate generic outputs or to address specific customer process.

The organisation of the documentation is also based on the workflow diagram breakdown. One particular kind of document (with related template) is defined for each workflow diagram super-state, in order to make the synthesis of modeling activities performed within this stage:

- **SND** (Stakeholder Needs Document) for Stakeholder needs definition stage
- **SyRD** (System Requirements Document) for Requirements analysis stage
- **SyDD** (System Design Document) for Logical and Physical architecture design
- **CND** (Components Needs Design) for Components needs definition stage

SysML artefacts and diagrams created when being in a given super-state of the workflow diagram are automatically attached with stereotypes indicating that they should appear in the document associated with this super-state. The names of these stereotypes are built with the name of artefact or diagram prefixed by the name of the target document (e.g: SND_requirement). It is also possible to manually apply documentation stereotypes when artefacts should appear in multiple documents



+02_a_System
+ContextToSystem_Specialization
+SystemRequirementsTraceabilityLinks
+Hybrid Vehicle (System)
+ElectricStationCommmunicationPort : Communica
+DiagnosisToolCommunicationPort : DiagnosisProt
+ZEVModeButton : On-Off signal
+AcceleratorPedal : Continuous signal
+StartVehicleButton : On-Off signal
+ElectricPlug : Current
+RoadContact : Torque
+BrakePedal : Continuous signal
+InformationDisplay : DisplayProtocol
+ : Hybrid Vehicle (System)
+Fct_StartVehicle
+Fct_AccelerateVehicle
+Fct_RecoverBrakingEnergy

Properties of 'InformationDisplay'

InformationDisplay

Full name: VALEO Methodology::02_a_System::Hybrid Vehicle (System).InformationDisplay

Type: FlowPort

Last changed on/by: 09/11/2011 16:52:42 VNET\Jpiques

Visibility: ● Public ○ Protected ○ Private ○ Package

Figure 05: Documentation Stereotype Example

The only thing left to do is to load in the publishing tool the pre-defined documentation template related to the workflow super-state to be documented, and then to launch documentation rendering. Diagrams and artefacts appearing in the final document are automatically filtered depending on their documentation stereotypes, i.e. on the stage of the workflow they have been created.

4. Coupling to requirement management tools

4.1. Efficient collaboration between tools

Speaking about requirements in general may lead to adopt wrong requirement management tooling solutions. In fact, initial needs are iteratively refined during the engineering process, producing different levels of so-called requirements, corresponding to very different kind of information. Typically these requirements can be classified in three categories:

- **User requirements** describe the expected services from the end user point of view.
- **System requirements** define the features of the system necessary to fulfil its mission.
- **Component requirements** specify the internal constitutive parts necessary to implement the expected features.

Therefore, believing that a unique tool has the capability to address efficiently these three layers of information is incorrect. On the contrary, a pragmatic approach adopted at Valeo is to take benefits from tools optimised for each field and to make them collaborate efficiently.

Another common mistake is to mix up two categories of requirements related tools:

- Requirement definition tools are containers of requirements (or any modeling artefacts used for specification).
- Requirement traceability tools do not define any requirements but have the ability to analyze requirements from requirement definition tools, and to analyze traceability links.

A tool of the second category (e.g. Reqtify) can therefore be used as a gateway to optimise collaboration between tools of the first category (DOORS, SysML Artisan Studio, Simulink, …), for synchronizing interface requirements and producing the whole traceability analysis. Another interesting property of this scheme is its ability to let people working with their discipline specific tools (such as Simulink for control design).

All the above mentioned principles are summarized on the figure below, showing the typical mapping of tools used at Valeo.

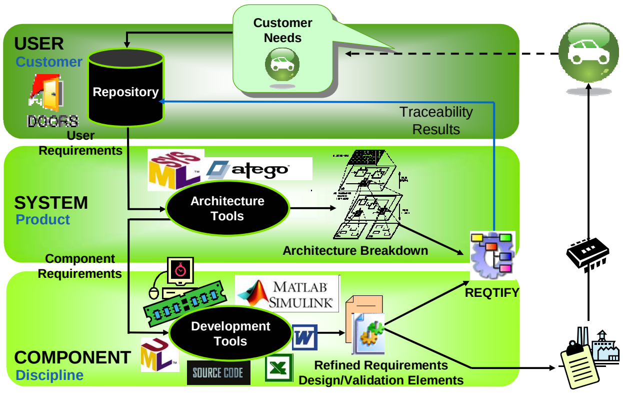


Figure 06: Requirements Related Tools Mapping

4.2. Distributed requirement storage

Classical requirement management approaches assume that all requirements shall be written in natural language inside a centralized database (typically DOORS). Then, SysML modeling artefacts are only considered as intermediary by-products that need to be finally translated into textual requirements. This process makes sense in the aerospace or railway transportation fields were certification procedures are document-centric by nature. However, in the automotive area, without any constraints from certification procedures, a pure model-centric approach is far more efficient.

Consequently, maximum benefits are taken from expressive power and semi-formal verification capability of the SysML modeling language. Requirements or requirements-like artefacts produced during system modeling activities are not reformulated in natural language into an external centralized database. On the contrary, the model itself becomes the reference and the automatically generated documentation only an illustration of this reference. This philosophy is also used at implementation level, where requirements or more exactly requirements-like artefacts remains embedded into discipline specific native models (such as Simulink models, for control design).

This approach optimises the requirement management effort because requirements are distributed among the tool locations where they have been defined, at each stage of the engineering process. As a counterpart, the consistency of the distributed requirements storage must be supported by powerful traceability tools, with efficient mechanisms for synchronizing interface requirements between modeling layers.

4.3. User requirements in external repositories

The initial stakeholder requirements (namely user requirements) remain captured in text specifications external to the SysML modeling tool, as in the classical approach. Typically, these specifications are stored in a DOORS database but may also be described using classical word processing or table editing softwares. The combination of the Reqtify gateway and of Artisan Studio modeling tool provides a mechanism to import external text requirements by creating mirroring SysML requirements directly into the SysML model and to later maintain these data synchronized. In fact, three kinds of synchronization mechanisms are available:

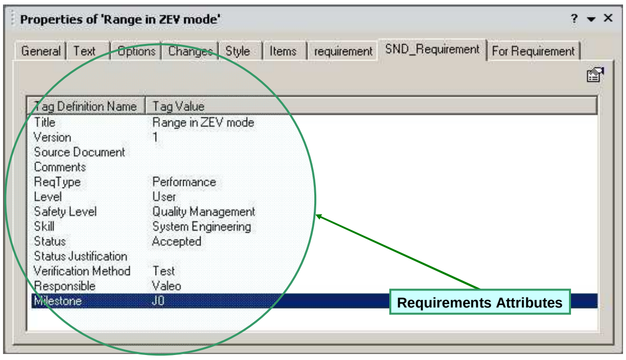
- Synchronization with a DOORS database
- Synchronization with any kind of requirement file captured with Reqtify
- Synchronization with Excel files (feature added by the Valeo Profile)

The SysCARS modeling activities performed to analyze stakeholder needs can result in proposing updates to the user requirements baseline. However, the textual requirements are formally updated and controlled in the external requirement repository and changes are propagated to the SysML model thanks to the synchronization mechanism

4.4. System and component requirements inside the SysML model

Requirements produced during SysML modeling activities are not reformulated in natural language into an external centralized repository. As a consequence, system and component level requirements are located inside the SysML model, taking benefits from internal traceability links with other model artefacts.

The standard SysML requirement object being mainly limited to an identifier and a description field, it has been necessary to add complementary attributes, for efficient requirement management. The figure bellows shows these additional fields added by the Valeo profile, using tag definitions.



Tag Definition Name	Tag Value
Title	Range in ZEV mode
Version	1
Source Document	
Comments	
ReqType	Performance
Level	User
Safety Level	Quality Management
Skill	System Engineering
Status	Accepted
Status Justification	
Verification Method	Test
Responsible	Valeo
Milestone	J0

Figure 07: Stereotyped Requirements Attributes

Another approach under investigation, but not used on the first pilot projects, is to limit the use of SysML requirements to non functional requirements. Then, functional requirements are represented by SysML artefacts attached to constitutive blocks, typically by operations and states. More than avoiding reformulating functions (described by operations) or states into functional requirements, this approach also saves the cost of declaring traceability relationships between structural elements and related functions. Indeed, operations and states are already tightly linked to their owning blocks.

4.5. SysCARS traceability model

The traceability model adopted in the SysCARS methodology has been pragmatically defined taking into account the features of the SysML modeling tool and the kind of verification that could be later performed.

The main rules used for defining traceability relationships are the following:

- **Derive** is used between two levels of requirements
- **Refine** is used between a use case or a scenario and the requirement elicitated
- **Satisfy** is used between a model artefact (state, port, operation, block) and the requirement implemented
- **Trace** is used between two representations of the same item, either refined between modeling levels or reformulated at the same level

The figure below represents the corresponding SysCARS traceability scheme.

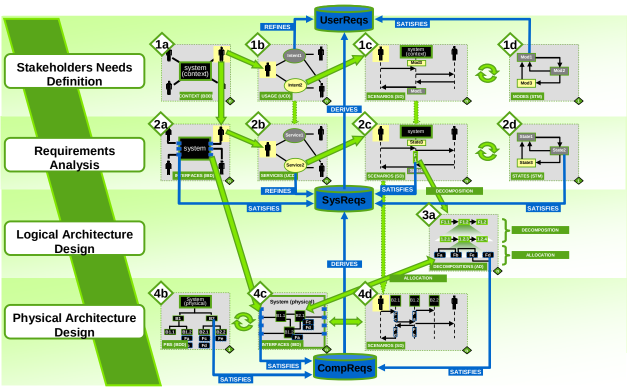


Figure 08: SysCARS Traceability Scheme

Refine and Satisfy relationships shall connect artefacts developed at the same modeling stage, while Derive and Trace relationships are also capable of linking artefacts of neighbouring levels.

4.6. Traceability analyses

The requirement traceability verification activities are invoked throughout the whole system engineering process. In fact, two kinds of traceability analyses are performed:

- Internal traceability analysis between SysML model artefacts, directly generated using the SysML tool,
- External traceability analysis, between the distributed requirement repositories, done using a general purpose requirement traceability tool such as Reqtify.

Internal traceability analyses are the ending activities performed at each stage of the workflow to verify the model consistency (refer to green states of the workflow diagram, [figure 02]). They use requirement tables and traceability matrices to check the coverage of all requirements by appropriate model artefacts, in accordance with the traceability model presented at the previous paragraph. These matrixes and tables are generated on demand at Excel format.

4.7. Towards modeling rules verification

In future projects, it is plan to use a modeling rule checker to automatically verify the consistency and the completeness of the model, in accordance with the traceability scheme. The idea of these rules is to check that each kind of requirement is effectively covered, and covered by the appropriate modeling artefact.

The verification rules will be based on several properties of the SysML objects and relationships:

- Category of object,
- Value of the stereotype indicating at which modeling stage the object has been produced,
- Values of its specific qualification attributes,
- Type of relationship used between objects.

A typical rule should be written under the following format:

- *“Each requirement of this level and of this type shall be covered by this category of object, with this type of relationship, the linked objects being respectively produced at these modeling stages”.*

5. Coupling to control design tools

The issue of coupling a SysML tool to discipline related tools (and particularly simulation tools) is not studied in general but limited to coupling to control design and software development environments, and particularly to Matlab/Simulink.

5.1. Static verification rather than co-simulation

Some approaches promote use of the SysML model as an integration framework for building a whole executable system model, in order to analyze the dynamics of the system. To support this, the static system modeling environment must be upgraded by execution mechanisms, with closed connection to discipline specific simulation tools.

This way has not been chosen at Valeo’s for several reasons:

- A higher degree of sophistication of the SysML environment would go against a wide adoption by (generalist) system engineers.
- Somehow, there is a contradiction between flat deep detailed modeling and the layered refinement approach promoted by system engineering.
- Simulation and co-simulation capabilities of SysML tools are quite limited compared to domain specific tools.
- For large scale system, a full integration simulated model is practically intractable.

The final objective being the verification and validation of the whole system model, a static verification of traceability properties, as discussed in previous paragraphs, has been preferred. The purpose is then to gain maximal confidence in the completeness of the intellectual progress which led to the physical architecture solution.

In a second time, as explained in the next paragraph, each component will be refined (and possibly simulated) independently in its discipline related development (and possibly modeling) environment, based on input data from the system model.

5.2. Transfer of structure description to Simulink

The problem of collaboration between SysML and Simulink is not stated in terms of (co)simulation but rather in terms of efficiently transferring and synchronizing modeling data between both environments. The synchronization at architecture description level was proven to be an efficient way to transfer information between system engineering teams and control design teams.

In practice, the approach selected for pilot projects was to transfer the IBD structural descriptions of control law components, from SysML towards Simulink. The resulting Simulink models, initially corresponding to empty structures are further refined, and control algorithms implemented, simulated and validated inside the Simulink modeling and execution environment.

The two figures below show an example of synchronization between a SysML Internal Block Diagram and Simulink dataflow model.

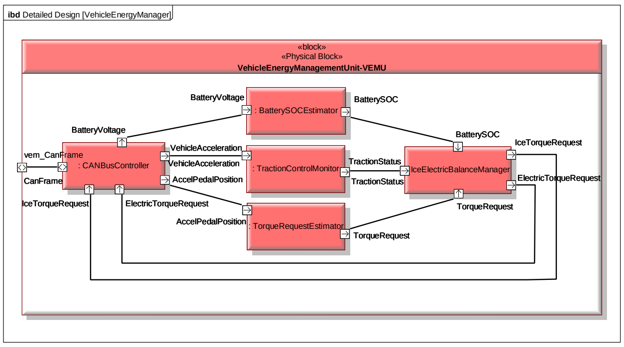


Figure 09: SysML Controller Architecture (IBD)

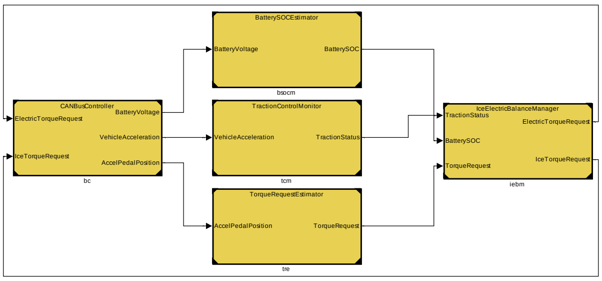


Figure 10: Simulink Controller Architecture (Dataflow)

At the end of control design activities, Simulink simulation results are summarized by measures of efficiencies (MoEs) finally attached as “values” to the corresponding SysML block.

Artisan Studio provides the main features required to synchronize and update efficiently SysML structural models and Simulink models: changes can be propagated in both directions. However, extensions in the existing mechanisms would be necessary for a full interoperability between both environments. These suggested evolutions are presented in the next paragraph.

5.3. Mapping between SysML and Simulink structural artefacts

The table below presents the detailed mapping for an efficient synchronization of structural descriptions between SysML Internal Block Diagrams and Simulink Dataflow models. Currently existing features of Artisan Studio are written in standard font, while suggested extensions are written with bold characters.

SysML	Simulink
Internal Block Diagram	Dataflow model
Block	System MDL
Part	Model Reference
	Sub-system
Flow port (in)	Inport
	Trigger port
Flow port (out)	Outport
Connector	Connecting line
Name of the connected out flow port	*Connecting line name (Signal name)*
Item flow	
Block description	*Documentation block*
Sequence diagram	*Signal builder*

Figure 11: Mapping Between IBD and Simulink

The main mandatory evolutions required are related to the ability to deal with Simulink events and not only with continuous flows. Indeed, events are systematically used to specify control flow mechanisms of algorithms. Therefore, it should be at least possible to map SysML (in) Flow Ports onto Simulink trigger ports (with “function call” trigger type option).

The ability to transfer names to Simulink flow lines is also mandatory, because in most situations they are used as variable names by automatic code generation tools.

It would be potentially very interesting to transfer data related to the expected behaviour of the algorithm. SysML sequence diagrams describing test cases could be translated into Simulink signal builder blocks.

The Simulink environment itself could also be improved with the capability to declare traceability links from Simulink sub-systems towards SysML artefacts, and particularly requirements. These links could be declared directly between tools or via an intermediate XMI file.

6. Functional safety handling with SysCARS

The new regulation ISO26262 focusing on Functional Safety, requires a higher level of formalization and traceability, and promotes the formalization of technical safety concepts in order to validate system architectures regarding safety expectations.

This part focuses on the ongoing SysCARS evolutions to support Safety In the Loop (SaIL).

6.1. General

System engineering shall reconcile all the different aspects of the system to be designed. Among the multiple points of view, Functional safety is a key one. The final architecture shall integrate both system and functional safety expectations. Therefore safety can not be addressed separately in a parallel and disconnected engineering domain.

Despite following a regular system engineering process, Functional safety uses dedicated analyses to achieve safety demonstrations. In the rest of the chapter, focus is dedicated to major synchronization points, exchanged artefacts and impacts regarding SysCARS process and tools.

6.2. System & safety process background

Performing process steps in the field, it appears that system engineering has a natural tendency to focus more on functional and nominal operations, while Functional safety focuses on malfunctioning and degraded operations. SysCARS is supporting both, and provides guidance for technical safety concept formalization as required by ISO26262:

At "Stakeholders Needs definition" level

Key engineering artefacts:

- specific scenarios related to critical safety contexts or degraded operations
- specific safety modes, related performance and availability of functions

SysCARS artefacts:

- using regular diagrams but dedicated to safety focus and interactions

Safety scenarios are a convenient way to capture in which conditions, malfunctions and safety goals are identified during Hazard analyses

At "Requirement analysis" level"

Key engineering artefacts:

- safety requirements refining previous understandings to define safety expectations
- specified external interfaces related to upper level safety mechanisms

SysCARS artefacts:

- using regular diagrams but dedicated to safety focus and interactions

Safety goals/requirements generally use a somehow negative form (e.g. for an Electronic Power Steering, avoidance of higher torque assist than requested). While system engineers are flowing down "positive" requirements (testable …), functional safety engineering consists in:

- transforming such "negative" goals into technical safety requirements allocated to implementation technologies (HW, SW, …)
- applying proven design patterns (e.g. safety mechanisms) during design of technical safety concept to achieve this transformation

Safety goals and requirements are implemented with regular SysML requirements and additional attributes (ASIL, related context …)

At "Logical architecture design" level"

Key engineering artefacts:

- breakdown of provided services into internal functions with flow down of ASIL

In general, due to a high level of reuse in the automotive projects, the logical architecture, as previously mentioned, is not seen as a valuable step, except for innovation projects with intermediate capitalization needs regarding allocation on multiple physical candidate architectures.

The same applies to safety, where the major objective is to ensure the link between high level requirements and final implementation. Furthermore the ASIL decomposition, having to demonstrate non interference, makes only sense taking into consideration hardware characteristics.

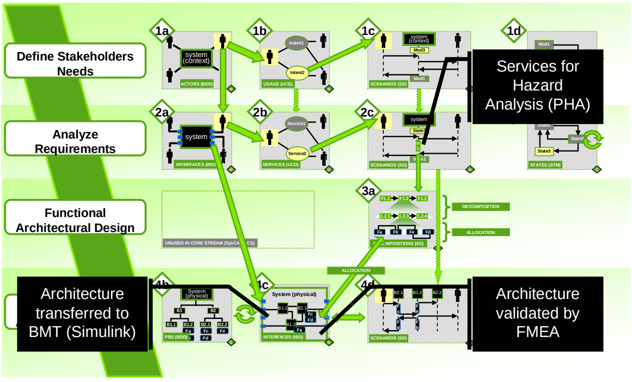


Figure 12: SysCARS-CS coupling to safety activities

At "Physical architecture design" level"

Key engineering artefacts:

- ASIL decomposition once internal functions are allocated to physical parts
- "ASIL" of internal parts and interfaces (due to highest ASIL function allocated)

- introduction of additional parts related to required safety mechanisms
- technical safety requirements impacting specific discipline

SysCARS artefacts:

- customization of IBDs with specific visual stereotypes to highlight safety related parts

In SysCARS-CS, iterations are achieved between functional decomposition and physical architecture to work out the relevant decomposition depth and ensure the mapping. The same process is used to iterate and introduce additional safety mechanisms.

Interactions and shared artefacts

System and safety points of view are completing each other, enriching the same artefacts which shall be kept consistent in a common referential. Considering process iterations, an initial set of scenarios, internal functions, physical parts … is released by system engineers, and then modified or completed by safety engineers to meet safety expectations. Setting up SysCARS as a common backbone to couple system and safety engineering solves these issues.

6.3. Safety specific verifications

Whereas system and safety generate common design artefacts, discrepancy occurs when taking into account safety related verifications.

Failure Modes and Effects Analyses (FMEA)

To check the safety relevance of a given architecture (functions allocated on physical), a common analysis is the FMEA. Each part is considered as possibly being faulty, and occurrence of undesirable events at system scope are assessed.

To achieve such analyses, the system descriptions shall be extended to dysfunctional modeling at part level:

- part, input and output fault modes
- part behaviour regarding faulty inputs, fault propagation to outputs
- intrinsic part fault propagation to outputs

These descriptions (together with part relations) allow mathematical checking of system properties such as occurrence of a undesirable event. Furthermore, working out dysfunctional behaviour per each part of the architecture allows:

- to enable computer aided verification of the architecture using FMEA principles
- to capitalize dysfunctional behaviour per part and therefore ease reuse of architecture subsets within new systems
- an easier peer reviewing of dysfunctional descriptions (per part), whereas review of traditional FMEA is difficult

Fault tree analyses (FTA)

While FMEA is a deductive approach, allowing to verify compliance of a given architecture regarding all undesirable events (bottom up), FTA is a top down approach (inductive) done per undesirable event working out relevant contributing faults.

Automated FMEA verification may output a FTA linking undesirable event to the relevant faults within the architecture (issues under work). This feature appears to be a major lever for efficient deployment of the ISO26262, while traditionally FTA and FMEA are concurrently done. Furthermore, the merged FTA is a key enabler to move forward quantitative analyses.

To draw FTA trees, either generated from architecture verification or done by hand from top to down, specific profiles have to be set up in the SysML editor.

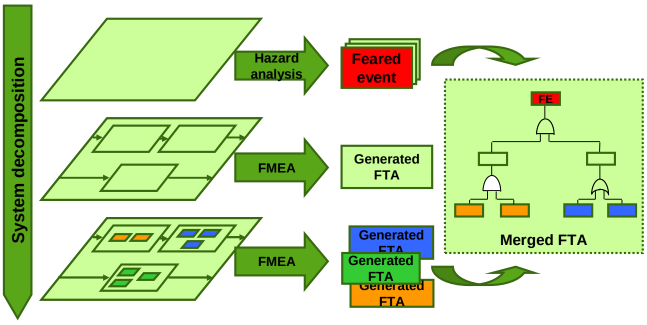


Figure 13: FTA Generation from SysCARS

6.4. Process and tooling considerations

From a process point of view, the key shared artefact is the physical architecture (key interface between system levels as well), including allocated technical functions, and completed with dysfunctional information.

Dysfunctional descriptions in system process

Such approach already exists in some industrial domains and tools are available in order to achieve such safety architecture verification (e.g. using the Altarica language and model checker).

These studies are in general performed by safety modeling experts. The involvement of system designer is reduced to providing information during interviews. This process and required safety modeling skills are a major show stopper for automotive deployment.

The SysCARS objective is to tightly couple engineering domains. System designers are the ones who best know both functional and dysfunctional behaviour. Therefore, dysfunctional modeling shall be integrated in their processes (at least for initial versions, later completed by safety experts).

Thus, a pragmatic dysfunctional formalization is under study, which will be a subset of Altarica available concepts, also taking benefits from York FTPC research and using as much as possible easy notations such as Boolean algebra.

Transformation to Altarica tools (or others) with generation of the required code (taking into consideration safety design patterns or functions typologies) is targeted.

6.5. SysCARS-SaIL status

While lessons learned have confirmed SysCARS capability to formalize ISO26262 technical safety concepts, extension is required to seamlessly implement “SaIL” concept (Safety In the Loop). Introduction of dysfunctional modeling and coupling to safety tools, will allow efficient safety architecture verification. Valeo internal effort is completed by collaboratively addressing these topics in the framework of European projects such as SAFE (ITEA2, Safe Automotive soFtware architEcture).

7. Conclusion

Learning on Valeo pilot projects have confirmed that the SysML language offers an adequate lever to extend the modeling practices to the area of System Engineering including functional safety analyses. Valeo experiences have shown that a successful approach requires a precisely defined modeling methodology (SysCARS).

Furthermore, the customisation of existing tools in a workflow driven mindset is mandatory. However, further improvements remain necessary on commercial tools regarding ergonomics and interfacing with simulation and safety analyses tools.

8. Acronyms

AD	Activity Diagram
ASIL	Automotive Safety Integrated Level
BDD	Block Definition Diagram
CND	Component Needs Document
FMEA	Failure Mode Effects Analysis
FTA	Fault Tree Analysis
FTPC	Fault Transformation and Propagation Calculation
GUI	Graphical User Interface
HW	HardWare
IBD	Internal Block Diagram
ISO26262	Automotive Functional Safety Regulation
ITEA	Information Technology for European Advance
MBSE	Model Based System Engineering
MDL	Simulink file extension
MoE	Measure Of Effectivness
REQ	REQuirement Diagram
SAFE	Safe Automotive soFtware architEcture
SD	Sequence Diagram
SND	Stakeholders’ Needs Document
STM	STate Machine diagram
SaIL	Safety In the Loop
SyDD	System Design Document
SyRD	System Requirements Document
UCD	Use case Diagram

9. References

[1] Eric Andrianarison, Jean-Denis Piques: *"SysML for embedded automotive Systems: a practical approach",* ERTS 2010.

[2] Françoise Caron: *"Exigences et ingénierie système : Mise en œuvre avec SysML",* EIRIS Conseil, 2008.

[3] A. Arnold, G. Point, A. Griffault, A. Rauzy “*The Altarica formalism for describing concurrent systems”,* 2000

[4] Richard F. Paige, Louis M. Rose, Xiaocheng Ge, Dimitrios S. Kolovos, and Phillip J. Brooke: “*FPTC: Automated Safety Analysis for Domain-Specific Languages”, 2008*

Further bibliographical references can be found in [1]