SysML for embedded automotive Systems: a practical approach

E Andrianarison, J-D Piques

SysML for embedded automotive Systems: a practical approach

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

1: VALEO Center for Electronics Excellence, 2 av. Fernand Pouillon , F-94042 Créteil Cedex
2: VALEO Engine and Electrical Systems, 14 avenue des Béguines, F-95892 Cery-Pontoise Cedex

Abstract: While SysML (System Modeling Language) is a leading topic for System Engineering (SE) in all domains, there is no pragmatic implementation of SE for automotive embedded systems and products. In this paper, a proposal is developed to meet the needs of Valeo product lines.

Keywords: Model Driven Engineering, System Modeling, SysML, System Engineering

1. Overview

As an introduction to the proposed SysML method, Part 2 is intended to emphasize major challenges related to system engineering in automotive and expectations from a model based approach. Methodological background and SysML positioning regarding research are developed in Part 3. Part 4 illustrates a SysML implementation of the model-driven SE process, experimented at Valeo's on pilot projects. Part 5 describes how SysML must be supported by an integrated set of tools. Remaining open issues are finally addressed in Part 6, to go further on the subject.

2. Motivations

2.1. Automotive challenges

While intrinsic systems complexity is often highlighted, the increasing complexity of organizations responsible for system development shall also be emphasized. The quest for the holly grail of R&D competitiveness and the globalization of the automotive industry have revealed a need for efficient use of worldwide competence centers and thus collaboration of teams located throughout the world. Thus, many (not saying all) automotive developments use multidisciplinary and distributed concurrent engineering.
To mitigate risks, automotive actors are deploying processes (CMMI, HIS-SPICE...) requiring more and more formalization and effort in document writing. Furthermore increasing complexity and safety expectations (upcoming ISO26262) are also in favor of that. Such trend has huge impacts on automotive actors who historically worked out a fine-grained trade off between cost, quality, performance and development cycles. Automotive lean processes have now to cope with higher formalization needs and still whilst ensuring flexibility and R&D efficiency. What is ultimately at stake is to improve formalization, avoiding administrative documentation drawbacks and degradation of expected gains.
At the same time, this context is also a unique opportunity while automotive industry is facing a major market breakthrough with a move towards electrical and hybrid vehicles. New products are expected and innovation is taking over usual carry over. From this point of view, system approach and higher formalization are key levers to secure investments and work out innovative systems and products.

2.2. Drawbacks of document centric approach

Implementation of requirement driven processes with document centric data management has been achieved in domains requiring high traceability of safety related designs. Taking benefit of their feedback, the following drawbacks have been identified:

- to achieve a minimum formal level with text, consistent effort is spent in writing valid requirements that are testable, clear, consistent ..
- engineering models such as state machines or architectures are difficult and expensive to transform into textual requirements
- consistency and configuration of these artifacts are hard to manage efficiently. This often leads to disconnect engineering artifacts and project documents

All these issues are impacting R&D efficiency and ability to achieve quick iterations, whereas adopting a model centric approach allows:

- to re-focus engineers on added value tasks by increasing actual time spent on design
- to ensure a seamless and costless traceability / consistency between different design elements (functional/organic architectures …)
- to provide graphical means which ease information sharing among different discipline teams

2.3. Benefits of model based approach

On top of that, major system engineering improvements are also expected. Sufficient effort shall be dedicated to problem statement and new means to work out and formalize stakeholders' needs are required. As different architectures and products are competing, support for design exploration and decision making is required.
The short coming ISO26262 regulation reinforcing functional safety activities is in favor of a tight coupling between system and safety processes.

Interfacing functional and dysfunctional descriptions is a prerequisite to achieve “Safety in the loop” (SaIL).
More generally, model based approach will provide a seamless and efficient way to rework designs in iterative innovation developments.

3. Methodological background

3.1. System engineering mindset

A common issue is to define what a system is, and when to use system engineering. Actually, the critical point is how you consider the “object” under design, not what the “object” actually is. Adopting a system approach, involves considering various aspects in a parallel and consistent way. These different focuses are commonly called points of view (viewpoints). These points of views may apply to system as well as components, parts …
Methods and thoughts mentioned here are not only targeting pure systems in Valeo but are also applicable to components/products, to ensure managed transition to the different implementation disciplines.

3.2. Point of views

Among research projects focusing on complex system modeling, SAGACE ([1], [2]) demonstrates that 9 “orthogonal” viewpoints allow a complete and optimized formalization of all system concerns.
From this theoretical basis, selection of viewpoints has been achieved taking into account current risks related to each concern on Valeo systems and products. For migration path reasons and to cope with system engineers’ learning curve, a first step focus is dedicated to the 6 (among 9) following viewpoints:

- External actors needs (static)
- Interactions scenarios/mission profiles
- Provided services to external actors
- Internal operations
- Physical resources and related breakdown
- Internal operations mapping onto resources

Most viewpoints related to system fairness regarding adaptation to context, actors, resources or services changes have been discarded.

3.3. EAST-ADL2 as automotive ontology

In order to make sure that models are meaningful, concepts used (functions, ECU, actuators…) and links shall be formally defined to provide an automotive ontology. During the last past 10 years, successive European automotive projects have contributed to work out EAST-ADL2 (Automotive Description Language, [3]) which defines an architecture framework and provides a metamodel of automotive specific artifacts.
Here also, a subset has been defined regarding Valeo specific needs as an automotive supplier:

- *FunctionalDesignArchitecture, FunctionModeling HardwareDesignArchitecture, Behavior,Requirement*
- *ErrorBehavior, SafetyRequirement*, *SafetyCase* to support the upcoming ISO26262

The related artifacts are inspiring the on-going definition of the Valeo SysML modeling to ensure a later model to model transformation (M2M) with EAST-ADL models. The following artifacts have been discarded for the first step: *Functional Analysis Architecture, Feature Modeling, Environment Modeling, Variability, Support, Verification Validation*.

3.4. SysML opportunity

Nevertheless, there is currently no modeling industrial tool available on the market using this automotive Domain Specific Language (DSL).
Being method agnostic and providing a very general boxology with “low” semantics, SysML therefore may be used to describe selected SAGACE system viewpoints and EAST-ADL2 artifacts. This allows for instance to use the same SysML block definition diagram (BDD) to describe either a functional or a physical breakdown. Thus SysML appears to be a unique opportunity to implement “EAST ADL2 friendly models” and take benefit from the defined architecture framework.
SysML method added value consists in:

- Selecting a diagram subset (among 9 available) which allows viewpoints and artifacts implementation in a convenient way for system engineers (learning curve optimization)
- Providing defined semantics to ensure diagrams meaning
- Providing an obvious diagram sequence which ensures modeling efficiency regarding company processes
- Taking into account interfaced processes and tools constraints such as Reqtify from Geensoft for requirement traceability or Simulink from The Mathworks for continuous modeling

The current method 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 listed in the bibliography ([4], [5], [6], [7], [8]), such as EIRIS method. This implementation is also taking maximum benefits from available features of the selected SysML tool for Valeo pilot projects, namely Artisan Studio from Atego.
Furthermore, for the first step, no particular emphasis was put on execution and simulation of the SysML model.

3.5. Valeo SysML example

Diagrams selected to illustrate the approach are extracted from a tutorial example used as training on pilot projects. For confidentiality and readability reasons it has been preferred to using diagrams directly extracted from pilot projects.

The method described below is under validation on pilot projects and is obviously subject to changes and improvements, to take into account field return.

4. Model-driven System Engineering process

The overall System Engineering process begins with the project context, considering the system to be developed as a black box, studying the environment, and then successively going deeper into the details. More precisely the System Engineering process is divided into four major phases:

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

In the following, the process and the sequence of activities will be described in a pure sequential way. However, in practice different steps could be performed simultaneously with iterative and mutual refinements. Moreover, even though not detailed in this paper, each phase systematically ends with a traceability analysis to check the completeness of activities performed and artifacts developed.
Finally, the kind of diagrams used at each step will be given by acronyms in brackets at title level: Block Definition Diagrams (BDD), Internal Block Diagrams (IBD), Use Case diagrams (UC), Sequence Diagrams (SD), STate Machine diagrams (STM), ACTivity Diagrams (ACT), REQuirements diagrams (REQ).

4.1. Stakeholder needs definition

Probably the most important step in a system development process is collecting initial needs to secure the goals that the system under development is to pursue.
The key steps of this phase are:

- Identify all the stakeholder needs
- Define the boundaries of the system and external actors involved
- Identify the user level operating modes
- Identify and describe the operational use cases
- Link the stakeholder requirements to the operational use cases

At this stage, all analyses are performed from the system external user point of view, the system being considered as a black box. The output of this phase is the Stakeholder Needs Document (SND) that makes a synthesis of all activities performed.

4.1.1. Stakeholder needs identification (REQ)

All individuals and organizations that may have an interest in the system are the potential source of requirements and therefore should be identified prior to all other activities. The key point is that stakeholder needs should describe the services expected by the system user, and not how the system will fulfill these needs.
The sources of stakeholder needs will be managed outside of the SysML model, within requirements documents or specific databases. It is particularly important to capture mission-level performance requirements and measures of effectiveness that will be used later to select the best candidate solutions.
The next step is to import stakeholder needs (with all their relevant fields) into mirroring SysML requirement objects (with same identifiers). A gateway mechanism such as Reqtify is required to perform a mono-directional synchronization (from external data to SysML) in case of change of source data. Because the standard SysML requirement format is quite limited, the extension mechanism of stereotypes is used to add new specific attributes to keep track of extra information contents. The first application of stereotyping attributes is to differentiate requirements imported from external requirement repositories from those created directly inside the SysML model. Moreover, other stereotyped tags should be declared to store specific information or resulting from the elicitation activities.

4.1.2. System context modeling (BDD)

The system context model represents the direct environment of the system and gives initial information about the system boundaries and the communication flowing between the system and external systems and users that the system interacts with.
The first step is to identify the different stages of the system lifecycle, from manufacturing to recycling. For each stage of the system lifecycle, one SysML block definition diagram is declared to model the operational context. The system itself appears in the center of the diagram as a single black box SysML block.

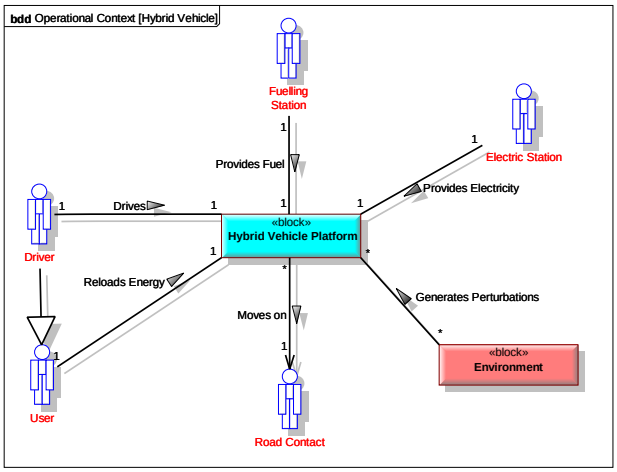

Figure 1: Operational context diagram

The next step consists in representing all currently known interaction partners, using SysML actor objects. An actor is not necessarily a concrete

individual or system, but a role played by an outside element in interaction.
Then, interactions between actors and the system are represented as SysML association relationships. The purpose is to identify basic information helpful to determine the services requested from the system embedded in its environment (the use cases), and not to give technical details of these services.
Even though, the context diagrams may seem obvious, in practice, searching for actors can lead to very fruitful discussions between the different stakeholders.

4.1.3. User modes identification (STM)

A mode characterizes a situation in the system life for which a specific expected behavior can be defined. It represents a state invariant of the system from the external user point of view, e.g. regarding the service given to the user and not how this service is performed by the system.
The goal is to derive a unique state machine aggregating all the modes and main transitions identified in the interactions with external users. Therefore, establishing the user mode state machine is an iterative process tightly coupled and interleaved with operational use case identification described in the next chapter.

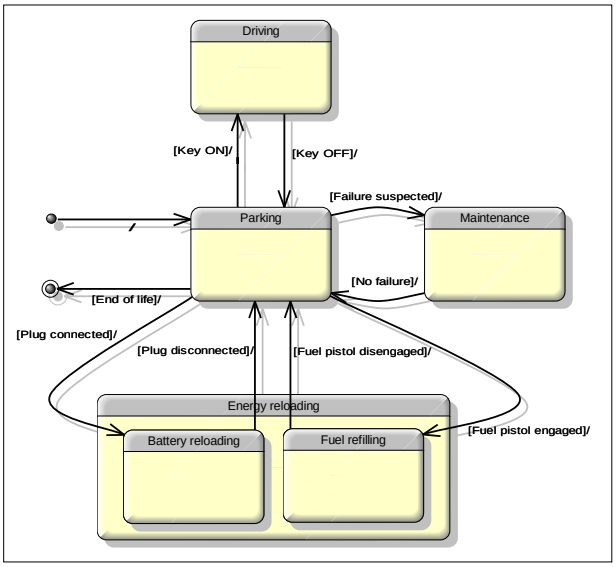

Figure 2: User modes state diagram

4.1.4. Operational use cases identification (UC, SD)

Use cases represent the services expected from the system, which means that they will be key elements to the requirement analysis stage. Indeed, use cases will help to refine stakeholder expectations and therefore identify system requirements in greater details.
Uses cases will be identified starting from the context model, asking what the actors want of the system, especially with regard to their roles and to incoming information flows. More precisely, a use case always refers to at least one actor; it is started by an external trigger; and it ends with a user result. Moreover, as many use case diagrams as stages of the system lifecycle will be described.

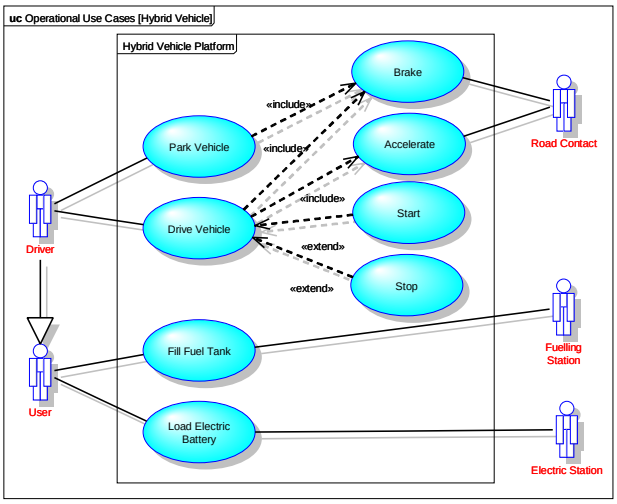

Figure 3: Use case diagram (user level)

In fact, a use case can be seen as a group of scenarios performed by the same main actor, with the same starting point and leading to the same ending point. These scenarios describe sequences of actions, beginning with the same pre-condition (trigger) and ending with the same post-condition (result); the pre-condition and the post-condition corresponding to modes in the user mode state machine mentioned in the previous chapter The purpose is to describe the sequence of steps involved in the different alternative scenarios, either leading to the expected result or to failure. This is done using SysML sequence diagrams.
It is particularly important to notice that, at this stage, scenarios will only be described at user interaction level. The sequence diagram established here is primarily aimed at identifying the system interfaces. It will be further refined at requirement analysis stage to identify functions performed by the system.

4.1.5. Traceability to stakeholder needs (REQ)

We remember that stakeholder (or initial) requirements refer to statements that define the expectations of the systems in terms of mission objectives, environment, constraints and measures of effectiveness/suitability, from the system user point of view. In order to make sure that all stakeholder needs are covered by the operational use cases, respective traceability links have to be established between SysML use case objects and requirements.

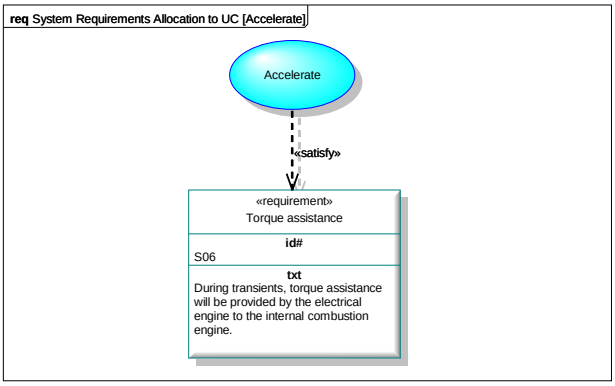

Figure 4: Requirement traceability diagram

4.2. Requirement analysis

The objective of the requirement analysis phase is to analyze the inputs previously collected, to move from a problem statement to an abstract solution.
The key steps of this phase are:

- Describe the interfaces of the system with external actors precisely
- Identify the system level operating states
- Develop and refine the system use cases
- Develop and refine the system requirements into external function and interface descriptions
- Link system functions and interfaces to the system requirements

At this stage all analyses are performed from system designer point of view, the system still being considered as a black box. The output of this phase is the System Requirement Document (SyRD), which summarizes all activities performed.

4.2.1. System interfaces description (IBD)

The objective of the system interface description phase is to give more details on the interaction flows between the actors and the system (always seen as a black box). The system physical external interfaces are described using internal block diagrams.

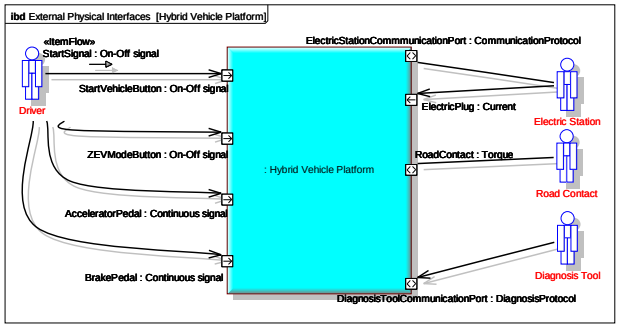

Figure 5: External interfaces description

To specify the kind of admissible data flow, a type indication shall be associated with each port, using SysML item types or flow specifications.
Several internal block diagrams are defined to describe the different contexts of use. Moreover, it is also possible to define several internal block diagrams for the same context of use, each diagram corresponding to a specific kind of interface (e.g.: mechanical, electrical, data processing buses…) to ease information sharing with involved disciplines.

4.2.2. System states identification (STM)

The objective of this step is to derive a unique state machine, resulting from the aggregation of all the states and main transitions identified in the system scenarios it participates in. This state machine will be the central element of the system model to which complementary artifacts will be later attached to, with the final target in mind being its validation by simulation.
The system state machine is not necessarily a refinement of the user mode state machine, as possibly new sub-states or suppressed states and even a completely different structure may be defined. Practically, establishing the system state machine is an iterative process tightly coupled and interleaved with system scenarios refinement described in the next chapter.

4.2.3. System scenario refinement (SD)

The objective of this stage is to refine user level scenarios in order to identify services to be provided by the system. Therefore, this activity is similar to a classical external functional analysis.
The sequence diagrams previously established to describe actors/system interactions are complemented by functions to be performed by the system (always seen as a black box). As shown on the figure below, these functions are denoted as loop interactions on the system lifeline and modeled as SysML operations attached to the system block.

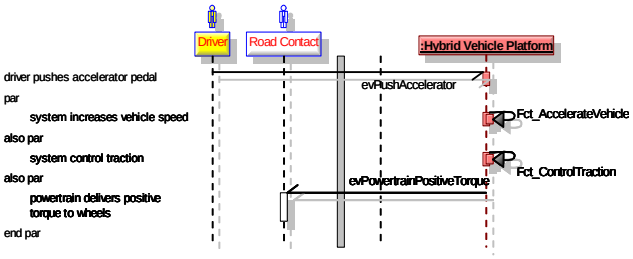

Figure 6: Scenario description (system level)

The refinement of user scenarios into system scenarios will possibly lead to new interaction events and even to a new structure of use cases.

4.2.4. System requirement capture

The process under validation in the pilot projects is to avoid a redundant writing of system requirements as textual requirements in an external requirement management repository, and to choose the SysML model artifacts as the reference for system and component requirements specification.
Indeed, interface requirements are modeled by ports and related type definitions, while functional

requirements are represented by operations, both being owned by the system block.

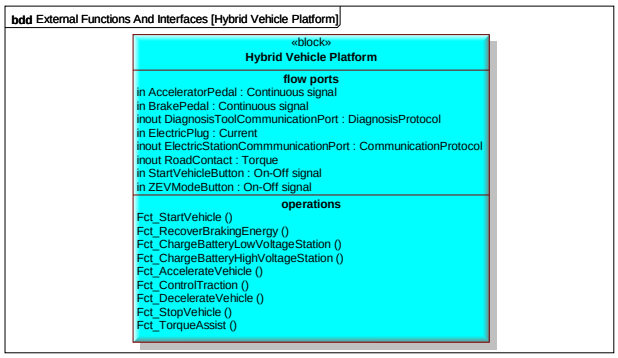

Figure 7: System requirements as block properties

4.3. Logical architecture design

The objective of the logical architecture design phase is to describe how the system will be internally structured to realize the expected features, but without any consideration of their upcoming physical implementation.

The key steps of this phase are:

- Identify the set of internal functions to be provided by the system
- Describe how these internal functions are activated depending on the system state
- Group these functions into coherent logical blocks
- Develop and refine the logical interfaces between the logical blocks
- Link internal functions and interfaces to the logical blocks
- Take into consideration any existing and relevant architecture

At this stage all analyses are performed from system internal point of view, the system being considered as a white box.

The modeling elements developed in this phase are included in the System Design Document (SyDD), which makes a synthesis of all logical and physical design solutions and their mapping.

4.3.1. Internal functions identification (ACT)

The objective of this step is to provide details on the internal behavior of system block operations. Therefore, the kind of task performed is similar to a classical internal functional analysis.

The activity diagrams use data flow and control flow representations in a hierarchical decomposition to work out internal activities that should be performed. The lowest level activities (namely leaves activities) of this hierarchy represent internal functions.

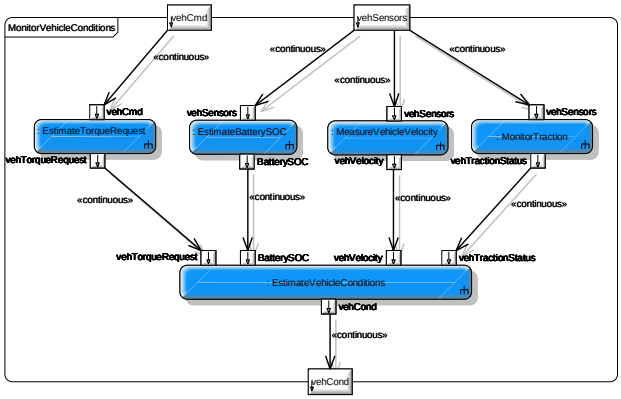

Figure 8: Lowest level activity diagram describing system internal functions

The key point is that the upper level activity diagrams describing the internal behavior are triggered by the system state machine, which will be an interesting and mandatory property for later execution of the system model.

4.3.2. Logical blocks definition

The objective of this step is to factorize and group system internal functions (leaves activities) into sub-sets to support the analysis of alternatives during physical architectural analysis. These subsets, called logical blocks, are abstractions of the components that will be implemented by the system; they perform the system functionality without imposing implementation constraints. The selection of each logical block is based on qualitative or quantitative criteria such as modularity or reuse of existing components.

The logical blocks are declared as SysML blocks with allocation relationships to the leaves activities performed. Moreover, operations are declared at block level to perform the activities which have been allocated to.

4.3.3. Logical architecture definition (BDD)

The logical architecture describes the compositional relationships between the upper level system block and constitutive logical blocks. It serves as an intermediate level of abstraction between the system requirements and the physical architecture. This intermediate level can reduce the impact of both requirements and technology changes on the physical design.

This description is performed using a block definition diagram.

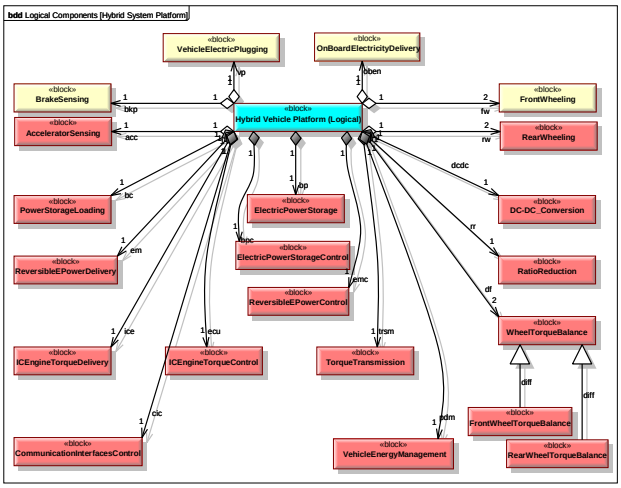

Figure 9: Logical architecture

4.3.4. *Logical internal interfaces description (IBD)*

The objective of the internal logical interface description step is to give more details on the interaction flows between the internal logical blocks.
The system logical internal interfaces are described using internal block diagrams. To specify the kind of admissible data flow, a type indication shall be associated with each port, using SysML item types or flow specifications.

4.4. Physical architecture design

The focus of the physical architecture design phase is on the development of a physical architecture (e.g. a set of components or component parts) capable of performing the internal functions required by the logical architecture.
The key steps of this phase are:

- Define relevant measures of effectiveness to identify candidate physical architectures to be investigated
- Investigate the most promising candidate solutions, and for each one:
  - Define a physical architecture capable of performing the required functions
  - Allocate the logical functions to physical components
  - Develop and refine the components physical interfaces and interactions
  - Develop and refine the components requirements
  - Evaluate the measures of effectiveness
- Select the best physical architecture solution based on measure of effectiveness criteria

The modeling elements developed in this phase are included in the System Design Document (SyDD), which makes a synthesis of all logical and physical design solutions.
The output of this phase is a also a set of Component Needs Documents (CND), which correspond to specifications for the components or disciplines modules to be implemented.

4.4.1. *Physical blocks definition*

The focus of the physical architecture design phase is on the allocation of logical operations to the components of a physical architectural structure. This structure may result from a previous trade study or a given (legacy) architecture.
The partitioning criteria used for allocation should reduce the impact of requirements and technology changes and more effectively address key issues such as performance, reliability, efficient re-use of COTS, maintainability, security and cost.
At the lowest level of the architectural decomposition, the functional allocation shall address the realization, e.g. which operation shall be implemented by which physical component developed by a single specific discipline (e.g. hardware, software, mechanics…).
The physical blocks are declared in the same way as logical blocks, using SysML blocks and linked with allocation relationships to the logical blocks they implement. Moreover, operations are implemented at block level to perform the activities which have allocated to.

4.4.2. *Physical architecture definition (BDD)*

The physical architecture describes the compositional relationships between the upper level (physical) system block and its constitutive physical blocks. This description is performed using a block definition diagram in the same way as for the logical architecture.

4.4.3. *Physical internal interfaces description (IBD)*

The objective of the internal physical interface description step is to provide more details on the interaction flows between the internal physical blocks, using internal block diagrams.

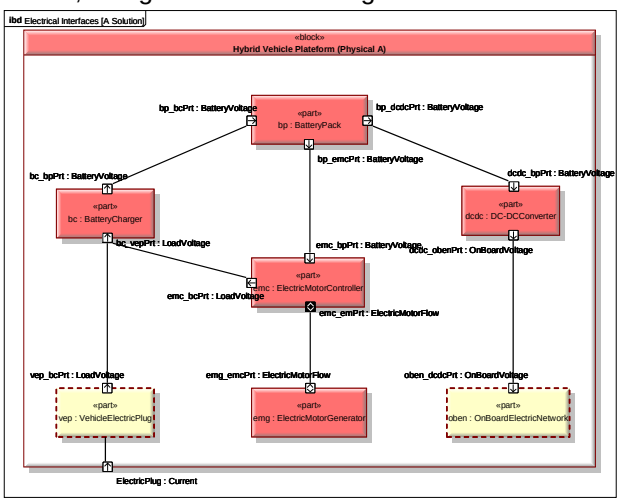

Figure 10: Physical internal interfaces description

To avoid information overload on the same diagram, several internal block diagrams will be described, each diagram corresponding to a specific kind of interface (ex: mechanical, electrical, data processing buses,…).

4.4.4. Component interactions definition (SD)

The focus of black-box sequence diagrams described at system level was on the identification of the required sequences of system functions (operations). Because some physical components may require significant refinement to address discipline-specific concerns and fully specify the related requirements, it may be necessary to establish white-box sequence diagrams focused on the collaboration between the different components. The derivation of white-box sequence diagrams is performed by refining the system level related sequence diagrams and splitting the black-box system lifeline into as many lifelines as constitutive physical components.
Moreover, a physical component may include a state machine as part of its specification, if it has significant state-based behavior.

4.4.5. Best physical architecture choice

Since there may be several competing hardware and/or software physical architectures that meet a given set of functional and performance requirements (e.g. the same logical architecture), several physical architecture alternatives may be investigated. The optimum physical design concept is selected based upon a set of measures of effectiveness (MoEs), that are weighted according to relative importance.

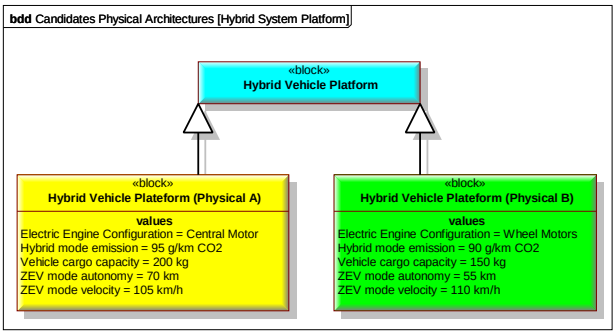

Figure 11: Physical candidate alternatives comparison

The estimations of MoEs result from specific engineering analyses performed with appropriate tools such as modeling and simulation environments and with different analysis objectives (performance, robustness, safety, cost…). The results from engineering analyses are therefore not elaborated using the SysML tool but incorporated back into the system model as value properties attached to the upper level system block describing the corresponding physical alternative.

4.4.6. Component Requirements Specification

The physical architecture model results in the specification of the components to be implemented by each specific discipline (e.g. hardware, software, mechanics…). The component specifications are typically captured as blocks with the appropriate black-box specification features attached as operations, ports or properties. The black-box component specification also includes functional requirements derived from scenarios analysis, and performance properties whose values (measures of effectiveness) have been determined through engineering analysis and trade studies.
The Component Needs Document (CND) describing the component requirements from the system designer point of view, is therefore automatically generated from the SysML model, through an extraction of all artifacts attached to the physical block under consideration.

5. Towards a systems development integrated environment

This chapter describes how the SysML system model must be supported by the development environment to provide an integrated framework for system development. A system development environment refers to the tools and repositories used for system engineering.

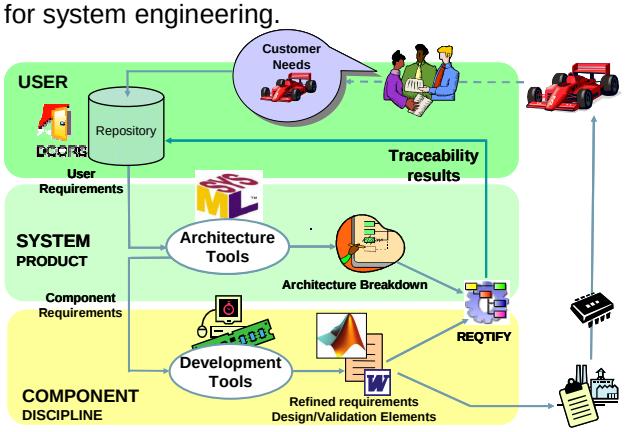

Figure 12: Tools for an integrated systems development environment

5.1. Requirement management and traceability

The requirement traceability management activity is invoked throughout the whole system engineering process, to establish traceability between the stakeholder requirements, the system model artifacts, and the system and components specifications. The stakeholder requirements are typically captured in text specifications external to the SysML modeling environment. The SysML modeling tool provides a mechanism to import text requirements by creating mirroring requirements directly into the SysML model and to maintain synchronization between the source requirements

and the corresponding SysML requirement objects. The analyses performed by the system modeling tool can result in proposing updates to the requirements baseline, but the textual requirements are formally updated and controlled in the requirement management tool.
Traceability reports are generated and used to analyze how the system design meets the system requirements and to perform impact analysis in case of changes. In fact, two kinds of analyses are performed:

- Internal traceability analysis, between the SysML model artifacts, directly generated using the SysML tool,
- External traceability analysis, between the text requirements repository and the SysML model boundary, performed, using a standard requirement traceability tool such as Reqtify.

5.2. Configuration management

Configuration management tools ensure that models and other development artifacts are maintained in a controlled fashion and that baselines are well identified.
The SysML tool is used with internal configuration management and change control features activated to provide a small granularity on configuration items manipulated. However, for major milestones, the packages contents of the SysML database are baselined and put under the standard configuration management tool. Therefore, packages are used to partition the model and constitute the unit of external configuration control.

5.3. System simulation

The SysML system model information can be used as a basis for building an executable system model to analyze the dynamics of the system. To support this, the static system modeling environment must be complemented by an execution environment. This execution environment could be either directly supported by the SysML tool or performed by an external simulation tool.
The approach selected for pilot projects is to transfer only components architecture description information to discipline specific tools in charge of refining an implementation dynamic model. For instance, internal blocks diagrams describing the software architecture of a control law are translated into Simulink dataflow models, with a direct mapping of blocks, flows and ports. The internal behavior of the component will be thereafter detailed and simulated in the Simulink modeling and execution environment.

5.4. Document generation

Document generation tools are used to prepare and manage formal documentation of the system design, in a format that is easily comprehensible by a broad range of stakeholders. These documents are an effective way to organize, validate and communicate system design information. Thanks to the 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. The only thing to do is to pre-define the expected document format (e.g. SyDD, CND…) by defining a specific template, which can be re-used on different projects, to generate documents related to project specific milestones.

5.5. Data exchange mechanisms

The interface between the system modeling tool and the discipline-related ones (e.g. hardware, software, mechanics…) is a critical issue. Indeed, the system modeling tool provides the component specifications for the different disciplines and it is crucial to avoid loss of information and reworking of data exchanged. Among the possible alternatives, approaches independent from the tools are preferred to those using tool-dependent interaction exchange protocol (for instance APIs). From this point of view, a file-based exchange mechanism based on neutral format or standard interchange format is a good answer.
Preferred relevant interchange standards are:

- RIF/ReqIF (Requirement Interchange Format) to exchange requirements between requirement management or traceability tools
- XMI (XML Metadata Interchange format) to exchange system models artifacts between SysML tools (with possible extension to other modeling and simulation tools)
- ISO AP-233 (Application Protocol 233) to transmit system engineering descriptions to domains or disciplines

Since the current level of maturity of these standards seems insufficient, confidence is put on short term improvements.

6. Lessons learned and perspectives

6.1. Boosting required cultural changes

As SysML method is implementing SE processes, it enables reinforcement of currently weak practices. For instance, deep understanding of the problem, before focusing on solution, is not currently natural and may be improved. The commonly used method to achieve external functional analysis APTE (APplication des Techniques d' Entreprise), may be improved as well. Moving to a use case driven approach taking into account interaction needs is a key improvement that SysML method naturally introduces but which may hurt current habits.

New ability to perform quick iterations between levels and viewpoints may worry project managers, used to progress indicators related to sequential process.

Therefore, training engineers is a key point, but achieving a mindset breakthrough definitely requires functional and project managers to be committed.

6.2. SysML adoption by engineers

While a DSL tool strictly focuses on users' needs, SysML tools provide confusing features and GUI which are unneeded. Whereas a wide range of users from system, hardware, software, mechanics disciplines are targeted, ergonomics and user friendliness is an issue for non software disciplines. SysML tool customization is therefore a key factor of success. SysML profiles or simply GUI simplification are definitely mandatory.
Different categories of users have been identified:

- Experts, defining SE methodology and SysML implementation. They are in charge of method and tool customization and evolution
- System leaders, having detailed understanding of implemented process. They are in charge of tailoring system modeling on a given project
- Contributors, having skills to model diagrams related to specific system engineering tasks
- Readers with reduced SysML knowledge, validating and verifying system description in the model

For efficiency reasons, each user category should model using a relevant specialized GUI depending on its skill level and task to be performed. At the moment, standard SysML tools are used by experts and pilot users, in order to tune SysML method and underlying data model. Based on field return, tool customizations will be specified and implemented in order to ensure wide adoption and user efficiency.

6.3. SysML modeling as central enabling brick

6.3.1. ISO26262 compliance

Upcoming ISO26262 regulation will require a higher level of formalization and traceability. At the interface between system and safety development teams, sharing of a common reference architecture is a key enabler. This helps avoiding system / safety concepts inconsistencies during design iterations.
Furthermore, what is expected is to share a common sound basis for both functional and dysfunctional studies and related simulations. While safety related effort will increase, automated safety analysis would help reduce costs impact.
This topic is covered by an internal project extension named "SaIL" (Safety In the Loop).

6.3.2. Coupling to behavior simulation tools

A key topic is the strategy for system model verification and validation and particularly model simulation. For the time being, SysML does not seem to be able to propose major breakthrough improvements regarding simulation; this topic will require deeper study. The current adopted strategy is a close collaboration with tools such as Simulink or Statemate to be able to execute the system model (or part of it) in the simulation environment. Still, there are open questions regarding the artifacts to be transferred and whether this is the most efficient approach.

6.3.3. Coupling to AUTOSAR authoring tools

As commonly admitted and formalized in EAST ADL2, wherever system and software architectures are different, a tight coupling shall be ensured in between. The M2M (Model to Model) related transformation need to be addressed. For such topics, improvements of standard interchange format (XMI, AP-233 …) are expected solutions.

7. Conclusion

SysML offers a promising solution to implement automotive ontology and extend current modeling scope. A practical approach and first results are given in the area of embedded automotive Systems and products. However, further iterations and improvements will be necessary to work out a final efficient and seamless process. This paper does not claim any theoretical novelty. Neither does it claim to be on the leading edge of SysML modeling compared to other domains.
However, it should enable sharing and collaboration with other industry experts, in charge of SysML implementation trade offs and facing similar issues regarding ergonomics and interfacing with simulation and safety analysis tools. It should help leverage synergies of cross-domain expertise and needs.

8. References

[1] C.Feliot : *"Modélisation de systèmes complexes: intégration et formalisation de modèles",* Thèse de l'Université de Lille I, 1997.

[2] J-M. Penalva: *"La modélisation par les systèmes en situations complexes",* Thèse de l'Université de Parix XI–Orsay, 1997.

[3] EAST-ADL2 Language and Profile http://www.atesst.org

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

[5] Sanford Friedenthal, Alan Moore, Rick Steiner: "*A Practical Guide to SysML - The Systems Modeling Language*", Morgan Kaufmann OMG Press, 2008.

[6] Tim Weilkiens: "*Systems Engineering with SysML/UML – Modeling, Analysis, Design",* Morgan Kaufmann OMG Press, 2007.

[7] Hans-Peter Hoffmann: "*Rational Harmony for Systems Engineering – Deskbook release 3.0*", IBM Software Group, 2008.

[8] Pascal Roques: "*SysML par l'exemple*", Eyrolles, 2009.