Adopting MBSE in Construction Equipment Industry: An Experience Report

Jagadish Suryadevara
Volvo Construction Equipment, Sweden
jagadish.suryadevara@volvo.com

Saurabh Tiwari
Mälardalen University, Sweden
saurabh.tiwari@mdh.se

***Abstract*—This paper is an experience report about introducing Model-based Systems Engineering (MBSE) at Volvo (Construction Equipment) and describes lessons learned. The recent growth in technologies such as electromobility, automation etc. in heavy construction machinery such as *loaders*, *haulers*, *excavators* etc. leads to increased complexity being addressed within embedded systems and software. Hence there is an increasing need for model-based development methodologies to facilitate flexible and distributed development scenarios, enhance communication among cross-functional teams, more importantly, traceability from requirements to system and software architectures. In this paper, we describe how the MBSE methodology was initially conceived, applied in an ongoing project, the challenges faced, and lessons learned. The paper also points to related works and future directions towards a holistic Model-Driven Development (MDD) framework.**

***Index Terms*—Model-Based Systems Engineering (MBSE), construction equipment industry, experience report, guidelines**

I. Background

Volvo (CE[1]) is a leading manufacturer of heavy construction machinery such as *Wheel Loaders*, *Articulated Haulers*, *Excavators*, etc. and is among top three in the corresponding market segment.

Over the past decade, the *mechanical aspects* within heavy equipment machines, are replaced with increasing amount of Electrical and Electronics. For instance, as shown in Fig. 1, hydraulic motors are being replaced with electric versions, new versions of the drive-line system are considered where electrified hub motors are introduced into wheels. The advanced technological changes described above in large complex products cause challenges for existing software development teams.

Besides the new functional aspects (e.g., machine features), the new *design concepts* lead to major changes in hardware and software. The traditional function-oriented development techniques are usually based on small incremental changes to existing software which is largely monolithic lacking the notions of "software architecture" and "system approach". These approaches increasingly lead to quality issues as well as maintainability, traceability problems. To manage the increasing complexity, instead model-based techniques such as MBSE (Model-based Systems Engineering) and MBD (Model-based Design) have gained industrial attention. However, application of these techniques at industrial scale requires established guiding principles, best practices, standardization, as well as evidence of tangible benefits for the stakeholders involved. It can be noted that the application of MBSE is based on the assumption of standard systems engineering practices in place (so called "document-based"). However lack of such standard practices leads to additional challenges while introducing "model-based" approach. Fortunately, there exist powerful modeling tools and methodologies to support introduction of MBSE approach, albeit with a steep learning curve. This paper describes challenges in adopting MBSE within large industrial contexts, such as Volvo.

[1]Construction Equipment AB, Sweden

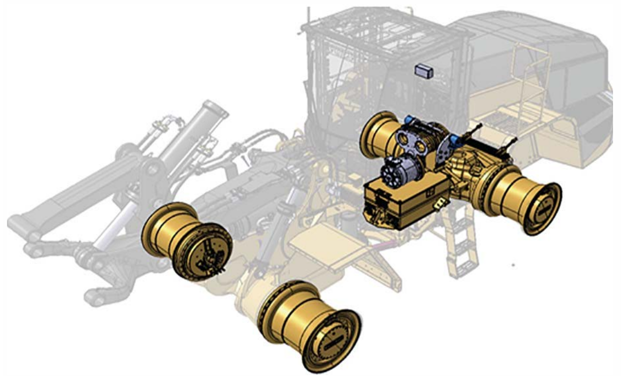


Fig. 1. Driveline system with Electrical motors in wheel hubs

II. Introduction

According to International Council on Systems Engineering (INCOSE[2]), "Model-based Systems Engineering (MBSE) is the *formalized application of modeling* as well as information management to support system requirements, design, analysis, verification and validation activities". The classical V-model way of developing systems require tool support to enable rigours and agile way of system development referred as *continuous development*.

The term "model-based" concerns the application of information modeling (visual and textual) methods, techniques and tools to SE activities (as defined by existing standards). For instance, the SE standard ISO 15288[3] related to systems engineering defines related processes and lifecycle phases. On the other hand, the standard ISO 42010[4] defines a framework

[2]https://www.incose.org/
[3]https://www.iso.org/standard/63711.html
[4]http://www.iso-architecture.org/ieee-1471/

for the formal description of the system as well as software architectures – in terms of *viewpoints* and *views* of the system and software aspects. The *viewpoints* correspond to the addressing stakeholder concerns that can be captured using relevant *views* (the views and viewpoints are further discussed later in the paper). Recently, new standards such as ISO 42020, and 42030 define enterprise level processes as well as process evaluation frameworks. These standards are aimed at emerging *System-of-Systems* (SoS) approaches and corresponding needs for enterprise level architecting methodologies e.g., service-oriented architectures.

In our view, the above described standards constitute the necessary foundations required for adoption of MBSE. For instance, at Volvo, the standard ISO 15288 has been adopted albeit with necessary customization required to reflect the organizational structure and corresponding responsibility interfaces. However there has been some gaps in formal documentation of "software architectures" as well as following truly "systems approach" within development process. Introduction of MBSE, provides and opportunity to address these gaps besides other benefits described previously.

This paper is a report describing initial experience gained in adopting MBSE approach at VCE, the experience of the system engineers towards MBSE, the lessons learned and challenges faced in applying the methodology. Besides, we attempt to evaluate (somewhat qualitatively) the effectiveness of the approach as well as overall maturity of the methodology based on the standards and the existing "best-practices" as described later in the paper.

The rest of the paper is organized as follows. In the next section, we present an overview of the state-of-practices about MBSE approaches. Based on this, we further arrive at a qualitative criteria to evaluate MBSE methodology described in this paper. Section IV describes the initial methodology i.e., the initial kick-off phase, planned work-flows etc. In Section V, we present an overview of the modeling techniques and patterns developed. In Section VI, we analyze the methodology on the basis of the validation criteria as well as the interviews conducted with various stakeholders. Finally, in Section VII, we make conclusions and point to some future works.

III. MBSE Overview: The State-of-Practice

In this section, we present an overview of the documented methodologies for MBSE.

According to Hallqvist et al. [1], MBSE is more than a mere technical task. It is essentially a "change process" affecting a very complex system where the important system elements are humans. Further, they rightly argue that adopting MBSE is not about learning a "modeling language" e.g., SysML, but more importantly addressing the stakeholders "concerns" during the change process. They rightly stated that the problem domain is valuable in defining common concepts and procedures, and further suggested that more time should be spent to build up a friendly infrastructure towards modeling, else MBSE practices become a burden, if not well defined until extensively used in the organization. In the rest of this section, we present an overview of MBSE methodologies and tools.

A. Methodologies & Tools: An Overview

Table I outlines the well-known methodologies for implementing MBSE. For large industrial contexts, these methodologies require customization to support the internal development processes and other organizational requirements. Besides these methodologies, there exist other approaches involving light-weight and comprehensive methods. For instance, Kass et al. [8] proposed a simplified MBSE schema based on traceability and maintainability. The rationale behind this schema is to avoid a steep learning curve for systems engineers. Spangelo et al. [9] reported their experience on applying domain-specific modeling techniques to model a standard CubeSat. They have tried to map the CubeSat terms with SysML concepts (SysML element and diagram types) and found that this improves the design and operation of CubeSat missions [10]. Hammarström et al. [11] reported the challenges encountered in the development of the Gripen Fighter aircraft, when combining the traditional SE with the MBSE methodology. They found that the identified system structure is not consistent with the domain structure, and thus employed domain analysis mechanism within the architecting process. Malone et al. [12], based on their experiences with MBSE at Boeing, reported that the MBSE helps in understanding the system architecture besides supporting SE processes. Friedenthal et al. [13] pioneered approaches to support MBSE using SysML, the standard system modeling language. Also, Bleakley et al. [14] proposed a methodology that enables users to explore the design space using SysML.

Tools are an integral part of MBSE methodologies and play significant role in the success or failure of the methodologies. There exist established tool frameworks e.g., Rhapsody (IBM Rational[5]), MagicDraw (NoMagic[6]), Integrity Modeler (PTC[7]) etc. in support of standard methodologies described earlier in this section. These tools provide a collaborative design, development and test environment for systems engineers and software engineers, using SysML/UML, the standard system and software modeling languages respectively. Other modeling tools like Modelio[8], Papyrus[9] also provide the UML/SysML environment for creating visual models. Roques et al. [15] describe the benefits of MBSE with the ARCADIA (Architecture Analysis and Design Integrated Approach) and the Capella[10] tool. Capella is a model-based solution deployed in a wide variety of industrial contexts. However, in spite of many modeling tools that exist today, there is lack of compatibility and interoperability that impede successful deployment of MBSE for large-scale development environment. To address this gap,

[5]https://www.ibm.com/
[6]https://www.nomagic.com/
[7]https://www.ptc.com/
[8]https://www.modelio.org/
[9]https://www.eclipse.org/papyrus/
[10]https://www.polarsys.org/capella/index.html

TABLE I
AN OVERVIEW OF MAIN MBSE METHODOLOGIES

S.No.	MBSE Methodology	Approach	Work-flows	Modeling Language	Tool Support
1	Harmony SE from IBM Telelogic [2]	Consistent with V model	Requirements analysis, system functional analysis, and design synthesis	SysML	Rhapsody TAU
2	OOSEM from INCOSE [3][4]	Consistent with V model	Analyze stakeholders needs, define systems requirements, define logical architecture, synthesize allocated architectures, optimize and evaluate alternatives, validate, and verify system	SysML	OMG SysML tools (integrated with other engineering tools)
3	RUP SE from IBM Rational [5]	Consistent with the spiral model	Inception, elaboration, construction, transition, and use case flow down activities	UML/SysML	Rational method composer with RUP SE plug-in
4	Vitech MBSE methodology [6] from Vitech Corporation	Concurrent design, incremental approach	Requirements analysis, behavior analysis, architecture/synthesis, and design V&V	System definition language (SDL)	CORE
5	INCOSE MBSE-based Agile Systems Engineering Framework [4][7]	Distributed teams, incremental approach	Analyze stakeholders needs, define systems requirements, define roles, define architecture, and verify system	-	-

the tool consortiums are moving towards a standard-based interoperability framework, namely Open Services for Lifecycle Collaboration (OSLC[11]). However, OSLC implementations are currently vendor-specific and far from providing a seamless integration of SE processes across wide tool spectrum (often multi-vendor) within large-scale industrial contexts, such as Volvo.

B. Validation Criteria

Reflecting on the MBSE overview presented in previous sections, we outline a qualitative evaluation criteria below, to be able to evaluate MBSE methodology described in the paper. Based on this criteria, a questionnaire is derived, as presented later in this paper, to interview the stakeholders and systems engineers involved in the application of the methodology. We state the criteria along four dimensions such as Organization, Methodology, Process and Tools.

Does the methodology,

1) ***Organization***
   - provide mechanisms to address the stakeholders concerns?
   - provide a collaborative environment for cross-functional domains/teams?
   - imply a steep learning curve for system engineers, project leaders?
   - provide a shared development platform, for both system and software teams?
2) ***Methodology***
   - support *reusability* of system/software artifacts?
   - provide *traceability* at system and software levels?
   - define easy to use work-flows as well as modeling patterns for complex tasks?
   - define guidelines and check-lists for modeling tasks?
   - define the modeling views and patterns w.r.t ISO 42010 framework?
   - support multiple development approaches such as water-fall, iterative, agile etc.?
   - support analysis techniques such as FTA / FMEA / Safety using models?
3) ***Processes***
   - supports main technical processes specified in ISO 15288?
   - provides enhanced requirement engineering?
   - provides architecture development techniques?
   - support verification and validation (V&V) of system / software models?
4) ***Tool support***
   - supported by existing tool frameworks?
   - supported by configuration management, change management capabilities (at model level)?
   - include simulation, code-generation support?
   - based on multi-tool environment with seem-less integration?

[11]https://open-services.net/

IV. MBSE: KICK-OFF PHASE

This initial phase was aligned with traditional pre-project activities, as the decision was taken to “try-out” the strategy within new “Pilot” project (to develop new generation of construction machinery with “electrified” subsystems as main focus). The outlined mission statement was “to deliver high quality Electronic Control System” for the corresponding product family. Several brainstorm sessions were organized among several key stakeholders as well as MBSE experts (as consultants) to outline an initial strategy for MBSE, albeit within a “limited scope”. Fig. 2 shows the “model context”

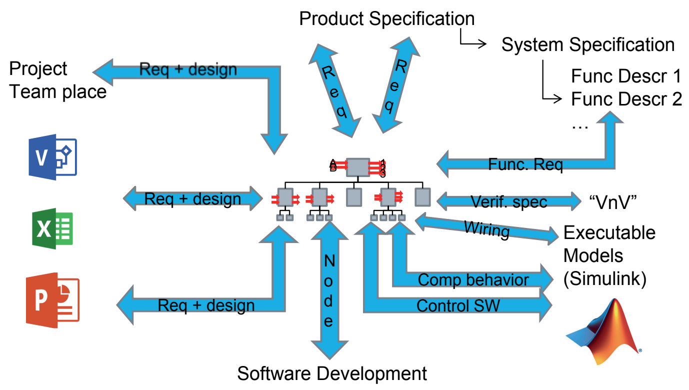


Fig. 2. Legacy Information "network" (illustration)

in the organization where there exist a variety of artifacts, spread across different tool contexts, some as the source of information while others are the target of overall information structure to be formally controlled, and change/configuration managed. The initial objectives of the MBSE strategy were limited to following:

1) Improve communication between the main "software" teams i.e., the Control system and Driveline System.
2) Improve system specifications and the functional requirements.
3) Enhance software verification and validation activities (system level).
4) Enhance traceability between System and Software levels and support analysis tasks (FTA [16], FMEA [17], Safety etc.)

Following activities were performed during this initial phase:

- Initial MBSE team was formed, with all relevant stakeholders besides new roles such as a modeling leader, modeler(s), model architect, and subject matter expert (consultant).
- Initial training sessions with focus on MBSE methodology, SysML modeling, tool (Rhapsody) conducted.
- A high-level methodology/work-flows defined (see Fig. 3)
- A set of guidelines describing "Modeling patterns" and checklists created to help with the modeling tasks.
- A scrum team setup initiated to coordinate the project tasks through sprints.

In spite of the "well-structured" setup described above, certain amount of "chaos" has kicked-in immediately after the start up of the project. This was essentially not due to "MBSE" but due to existing "information network" (see Fig. 2) over several sources as well as heterogeneous tool environments. This "network" represents legacy information artifacts often inconsistent, as well duplicated. However, these impediments were to be "expected" though the initial time-plan regarding developing the models and review process (ref. Fig. 3) were often not met.

Some of other challenges faced during the kickoff phase as below:

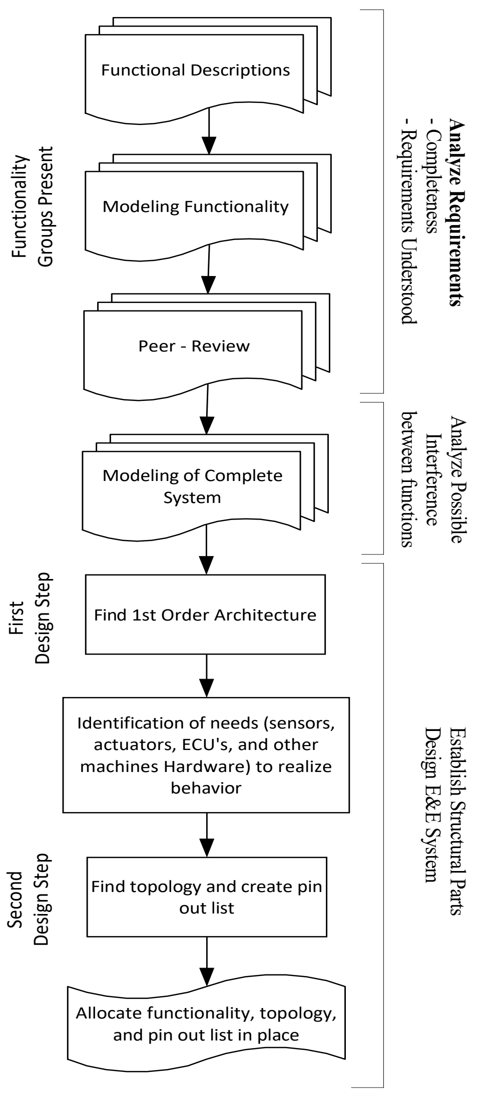

Functional Descriptions
Modeling Functionality
Peer - Review
Modeling of Complete System
Find 1st Order Architecture
Identification of needs (sensors, actuators, ECU's, and other machines Hardware) to realize behavior
Find topology and create pin out list
Allocate functionality, topology, and pin out list in place
Functionality Groups Present
First Design Step
Second Design Step
Analyze Requirements
- Completeness
- Requirements Understood
Analyze Possible Interference between functions
Establish Structural Parts Design E&E System

Fig. 3. The adopted work-flow for MBSE implementation.

- The choice of SysML diagrams for specific modeling tasks was not clear often leading to long discussions and "unscheduled" brainstorm sessions.
- Existing "guidelines" were not sufficiently adequate and instead in-house development of "guidelines" and "patterns" was identified.
- The tooling environment was not considered very "friendly" (partly due to inexperience working with the tool)
- Collaborative environment of the tool was not very efficient.

However, after the initial impediments, the progress had slowly settled-in, towards a stable work-flow (Fig. 3) as well as the methodology development. In the next section, we describe the methodology in detail.

V. Methodology Description

The main work-flow as shown in Fig. 3, consisting of several modeling tasks, can be divided into two phases corresponding

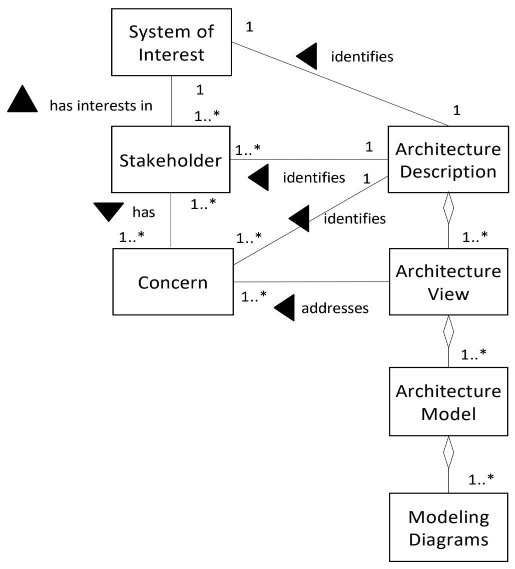


Fig. 4. Conceptual model (Ref. ISO 42010)

to Systems Engineering and Software Engineering. These phases correspond to creating necessary architectural "views" (refer Fig. 4) such as logical, functional, software, as well as hardware. These views correspond to corresponding stakeholder "concerns". Further "modeling patterns" are developed to capture these views.

The overall objectives of MBSE methodology, as outlined in previous section, is to identify "quality" requirements as well as transform requirements into initial design (i.e., architecture development), referred as logical architecture, as well as develop system and software functional architectures. The proposed SysML-based methodology (Fig. 5) is the main vehicle to achieve the objectives.

A. *Systems Engineering Phase*

The corresponding tasks (as per the main work-flow described earlier) as below:

Analyze requirements: Based on the existing documents (e.g., system specifications), the requirements are analyzed and modeled. The specific modeling pattern is based on "requirements modeling", "use case modeling" if necessary complemented with "sequence diagrams".

Develop logical & functional breakdown structures: Both logical and functional architectures are created in parallel (referred as the "1st Order Architecture" in the workflow). While the logical architecture refers to the "problem domain" i.e., external to the wheel-loader context, the high-level functional breakdown structure refers to the "solution domain" i.e., current system designs, and paves the way for detailed system architectures, incremental fashion. It may be noted that these first-cut architectures also correspond to the organizational entities at VCE. This is a very pragmatic approach, as well facilitate identifying the organizational interfaces and division of responsibilities (thus central to systems engineering processes). The modeling pattern adopted was based on SysML BDD (Block Definition Diagram) and IBD (Internal Block Diagram). Statemachines too were employed to capture the functional behavior (also incrementally).

Analyze functions: The task is performed in two phases. Initially functions are analyzed in isolation. This step also includes identifying the functional requirements. After all the functions (or at least a collection of related functions) are analyzed, the "models" corresponding to these functions are "combined". This phase is further described in the next task below.

System-of-Interest view: In this work, the System-of-Interest approach is employed in a restricted sense, with reference to break-down structures described above. For instance, the overall functional behavior of a System-of-Interest (SoI) is captured in a piece-meal fashion (described in previous step above) giving plenty of opportunity for communication between different stakeholders. Also, an opportunity to evaluate the *function* under consideration away from rest of the functionality. This paves the way for quality functional requirements to be verified and validated against implemented *design concepts* as well as the corresponding software.

System Specifications: The combined functional view of a SoI, developed in incremental steps, as a system specification document. The corresponding modeling pattern consists mainly of a state machine. These constitute the "white-box" view of the system corresponding to the "black-box" view w.r.t the system architecture described above.

B. *Software Engineering Phase*

In this phase, a system (e.g., the main Control System, Drive line System etc.) is implemented largely in Software. However, the "Software Parts" are identified during the realization view. Further, the "Software Architecture" views are developed to support allocation of software parts to ECUs (besides sensors, and actuators, also part of the hardware view).

Realization Views: These views correspond to actual conceptual design in terms of hardware and software allocations as well the most important design decisions. Also, the modeling views correspond to methodological interface between Systems Engineering (earlier views described above) and Software Engineering domains (e.g., Simulink models). The SysML *operations* within a system specification are "divided" into *control* or "intelligence" parts (i.e., Software) alt. non-*control* parts corresponding to hardware implementation.

Software Views: This is the final phase of the SysML-modeling activity and concerns the software architecture modeling. The models in the previous sections play an important role in identifying the "software" parts from "hardware" parts. However, certain amount of domain expertise is needed besides the technical trade-offs to be made regarding whether a certain SysML "Operation" is to be implemented in software or hardware (e.g., sensor).

An example of the SysML diagrams corresponding to the work-flow and the views described above are presented in the Appendix Section.

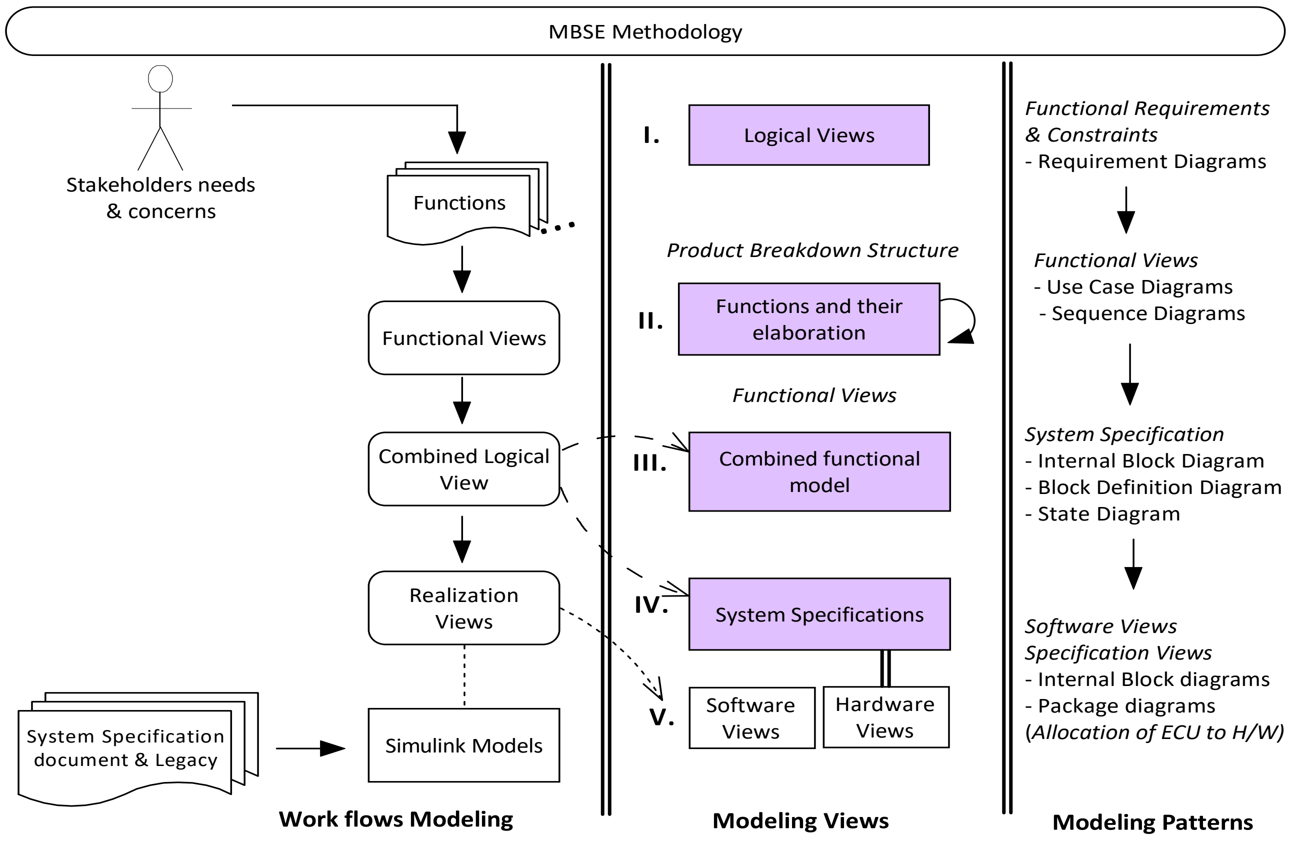


Fig. 5. Methodology Overview: Architecture & Modeling Patterns

VI. Methodology Validation

In this section, we analyze the proposed MBSE methodology on the basis of the validation criteria described in Section III-B. Four stakeholders (one interview per role) such as system engineers, function leaders, architects, and verification leaders were interviewed. The responses were collected on the four-point Likert scale (Strongly Agree (SA); Agree (A); Disagree (D); Strongly Disagree (SD)). Additionally, we recorded their general experiences as well as the lessons learned with the MBSE methodology.

Table II shows the responses collected across the four dimensions i.e., organization, methodology, process, as well tool support. This qualitative evaluation revealed that the methodology

1) provides sufficient work-flows e.g., structured reviews, modeling views to address the stakeholder concerns.
2) provides a “collaborative environment” for cross-functional teams.
3) provide a shared “development platform”, for both system and software teams.
4) thanks to inherent object-oriented features, support reusability of system/software artifacts (models).
5) provide traceability across system and software levels within single tool environment though, i.e Rhapsody.
6) define easy to use work-flows as well as modeling “patterns” to simplify complex modeling tasks for systems engineers not experienced in modeling.
7) define guidelines and checklists for modeling tasks to ensure overall “quality” and consistency of the models.
8) provides enhanced requirement engineering process, thanks to cross-functional teams and collaborative environment.
9) provides architecture development techniques paving the way for improved specification of “interfaces” as well “interface Requirements”. This further enhances system integration and verification aspects.

However, some shortcomings and limitations of the methodology too identified, as below.

- In spite of the initial training sessions and standing support from methodology expert, the methodology did imply a steep learning curve. This was partly due the initial training largely focused on learning “SysML” instead more important methodology aspects such as architecture development.
- Lacks support for analysis techniques such as FTA / FMEA / Safety using new artifacts i.e., models (this points to need for integrating corresponding “domain ontologies” into the methodology as part of future work).
- The major shortcoming of the methodology was insufficient V&V activities. The reasons were two-fold; lack of model-based testing (MBT) knowledge within the team. The second reason was lack of tool integration between Rhapsody and existing testing environment.
- The scope of the methodology was limited to architecture

TABLE II
EVALUATION RESULTS [STRONGLY AGREE (SA); AGREE (A); DISAGREE (D); STRONGLY DISAGREE (SD)]

Dimension	Criteria	System Engineer	Function Leader	Architect	Verification Leader
Organisation	Does the methodology provide mechanisms to address the stakeholders "concerns"?	SA	SA	SA	SA
	Does the methodology provide a collaborative environment for cross-functional domains/teams?	A	A	A	A
	Does the methodology imply a steep learning curve for system engineers, project leaders?	SA	SA	SA	SA
	Does the methodology provide a shared development platform, for both system and software teams?	SA	A	A	A
Methodology	Does the methodology support reusability of system/software artifacts (models)?	A	SA	A	A
	Does the methodology provide traceability across system and software levels?	A	A	A	A
	Does the methodology define easy to use work-flows as well as modeling patterns for complex tasks?	A	A	A	A
	Does the methodology define guidelines and check-lists for modeling tasks?	SA	SA	SA	SA
	Does the methodology define the modeling views and patterns w.r.t ISO 42010 framework?	A	A	A	A
	Does the methodology support multiple development approaches such as waterfall, iterative, agile etc.?	A	A	A	A
	Does the methodology support analysis techniques such as FTA/FMEA/Safety using models?	SD	D	D	D
Processes	Does the methodology supports main technical processes specified in ISO 15288?	A	A	A	A
	Does the methodology define the modeling views and patterns w.r.t ISO 42010 framework?	SA	SA	SA	SA
	Does the methodology provides enhanced requirement engineering?	A	A	A	A
	Does the methodology provides architecture development techniques?	A	A	A	A
	Does the methodology support verification and validation (V&V) of system/software models?	D	D	D	SD
Tool Support	Is the methodology supported by existing tool frameworks?	D	D	D	D
	Is the methodology supported by configuration management, change management capabilities (at model level) of existing tools?	D	D	SD	SD
	Does the methodology include simulation, code-generation support?	D	D	D	D
	Does the methodology based on multi-tool environment with seem-less integration?	D	SD	D	SD

modeling, and not integrated well with general tool capabilities available e.g., simulation and code generation.

- Lack of tool integration, and thus existing processes, was a major factor in limiting the benefits. For instance, possibility of OSLC implementation in existing tools investigated (e.g., to integrate requirements with existing test practices) but not succeed due several factors including budget over-runs.

In nutshell, the methodology was largely "successful" and results "satisfactory" as far as the defined scope and stated objectives are considered. In the next section, we describe some general lessons learned.

A. Lessons Learned

The major lessons learned are as follows:

1) The modeling language SysML is a major factor in the methodology. However, MBSE concerns creating an optimal set of sufficient models corresponding to existing "document-based" artifacts. The latter task is beyond the

scope of a single project and requires sufficient resources and commitment from the organization.

2) In spite of long-term objectives with MBSE, some short time benefits can be easily obtained. For instance, communication between cross-functional teams, single source of information, traceability between requirements and system artifacts etc. Project planning need take this into account and commit sufficient resources in advance avoiding budget over-runs later.
3) MBSE does imply heavy investment as well as a steep learning curve, especially in the initial phases, and is beyond the scope of a single project. Fortunately, this was well recognized by both the stakeholders as well as the system engineers.
4) The Tool interoperability plays a major role to the organizational level success of MBSE. To this extent, the successful implementations of OSLC standard within major tools is awaited eagerly, especially in the context of integrating legacy tools.
5) One major area of concern, is how to integrate non-model based artifacts within MBSE framework. This is particularly important in view of organizational units that prefer to keep SE practices document-based.

VII. Conclusions and Future Work

Hallqvist et al. [1] describe a phased approach in introducing MBSE in large-scale industrial contexts. Their MBSE journey had begun with a pre-study aimed at modeling a part of an existing system, gradually extended to modeling a complete system. The overall approach divided into addressing specific systems engineering processes i.e., Requirements Management, Product Development, and System Design. Instead of Requirements engineering process, the first phase focused on architecture development process addressing identification of system interfaces and the design process for modeling the logical architecture.

Malone et al. [12] at Boeing introduced MBSE for the purpose of verification and validation of the large-scale system. By executing a well-formed set of model analysis queries, it was possible to identify both modeling as well as the specification errors. Bonnet et al. [18] at Thales Group shared their four years of MBSE experience within the organization, identifying the stakeholders responsible for this cultural changes and the system engineers who play the critical role in this change.

The above described large-scale industrial experiences emphasize that adoption of MBSE cannot be achieved in one go. Besides the technical aspects of developing the models, it requires a cultural change which affects the overall organization structure in terms of adopting new ways of working integrated with system engineering processes.

The MBSE methodology described in this paper, mainly focuses on four different processes, as shown in Fig. 6, namely *Requirements Analysis, Architecture Design, Implementation*, and *Integration Process* with the purpose of introducing modeling to the system engineers. However, the requirements engineering was indirectly supported. The main emphasis was the identification of functional interfaces, modeling the software and hardware architectures of the system as well enable allocation of logical aspects to physical architectures (to support traceability). However, the verification and validation processes have not been explored, as it was not clear how to extend/integrate current non-model based artifacts e.g., test-cases etc to model artifacts. This would be the focus of next phase of MBSE activity.

The MBSE methodology described in this paper was primarily a reverse-engineering effort in modeling the functionality from legacy architectures and implementations. However, the work flows, architectural views, modeling patterns etc. are generic and also applicable in truly top-down approach. Thus the methodology developed follows the general principles and guidelines of SE practices.

Adopting MBSE methodology for a complex industrial context is a long drawn process, gaining maturity over several projects. Currently, an extended version of the methodology is being developed with support from a major tool provider (PTC). The Integrity tool chain from PTC, consisting of Lifecycle Manager, Modeler as well as Windchill, with seemless tool integration facilitated by the OSLC standard, will be a major enabling factor towards a holistic MBSE methodology (integrating both ALM (Application Lifecycle Management) as well as PLM (Product Lifecycle Management) domains).

Acknowledgement

This work is partially funded from the Electronic Component Systems for European Leadership Joint Undertaking (ECSEL-JU) under grant agreement No. 737494 and The Swedish Innovation Agency, Vinnova (MegaM@Rt2).

References

[1] J. Hallqvist and J. Larsson, "Introducing MBSE by using systems engineering principles," in *INCOSE International Symposium*, vol. 26, no. 1, Wiley Online Library, pp. 512–525, 2016.

[2] J. Estefan, "Survey of model-based systems engineering (MBSE) methodologies," *Incose MBSE Focus Group*, vol. 25, no. 8, pp. 1–12, 2007.

[3] J. Estefan, "MBSE methodology survey," *INSIGHT, Wiley Online Library*, vol. 12, no. 4, pp. 16–18, 2015.

[4] S. Friedenthal, R. Griego, and M. Sampson, "INCOSE model based systems engineering (MBSE) initiative," in *INCOSE 2007 Symposium*, 2007.

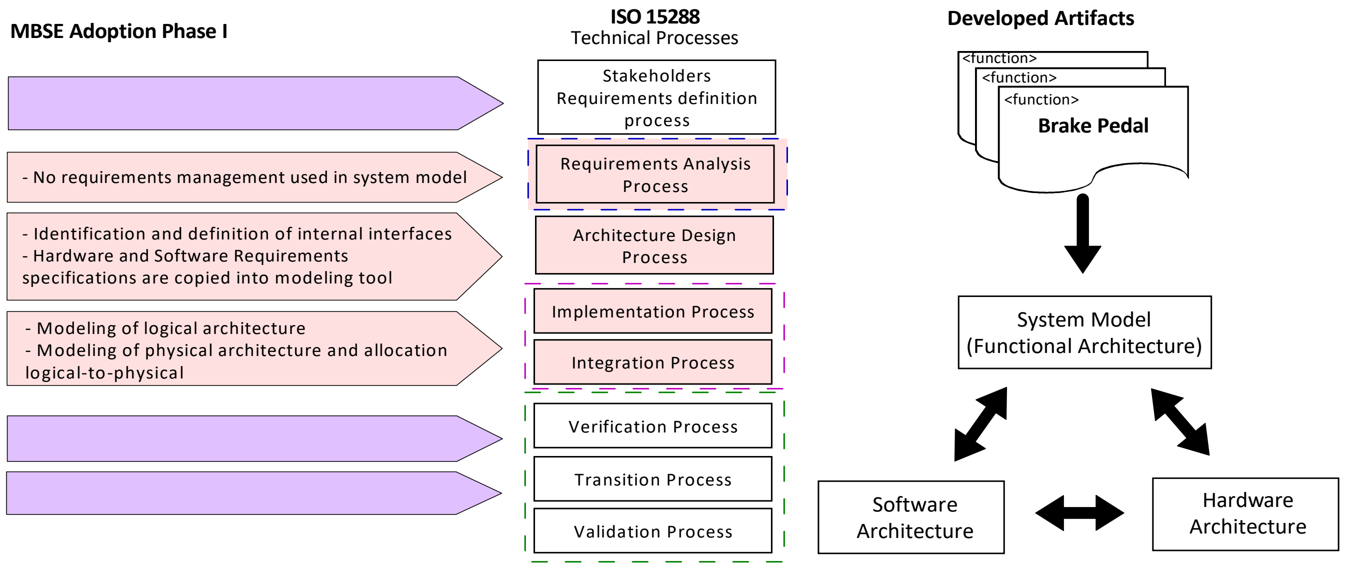
[5] M. Cantor and R. Plug, "Rational unified process for systems engineering part 1: Introducing RUP se version 2.0," *The Rational Edge (August 2003)*, 2003.

[6] A. L. Ramos, J. V. Ferreira, and J. Barcel, "Model-based systems engineering: An emerging approach for modern systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 1, pp. 101–111, Jan 2012.

[7] B. Schindel and R. Dove, "Introduction to the agile systems engineering life cycle MBSE pattern," *INCOSE International Symposium*, vol. 26, no. 1, pp. 725–742, 2016.

[8] N. Kass and J. Kolozs, "Getting started with MBSE in product development," *INCOSE International Symposium*, vol. 26, no. 1, pp. 526–541, 2016.

[9] S. C. Spangelo, D. Kaslow, C. Delp, B. Cole, L. Anderson, E. Fosse, B. S. Gilbert, L. Hartman, T. Kahn, and J. Cutler, "Applying model based systems engineering (MBSE) to a standard CubeSat," in *2012 IEEE Aerospace Conference*, March 2012, pp. 1–20, 2012.



MBSE Adoption Phase I
ISO 15288
Technical Processes
Developed Artifacts
Stakeholders Requirements definition process
- No requirements management used in system model
Requirements Analysis Process
- Identification and definition of internal interfaces
- Hardware and Software Requirements specifications are copied into modeling tool
Architecture Design Process
- Modeling of logical architecture
- Modeling of physical architecture and allocation logical-to-physical
Implementation Process
Integration Process
Verification Process
Transition Process
Validation Process
<function>
<function>
<function>
Brake Pedal
System Model (Functional Architecture)
Software Architecture
Hardware Architecture

Fig. 6. Systems Engineering processes and corresponding modeling artifacts.

[10] D. Kaslow, L. Anderson, S. Asundi, B. Ayres, C. Iwata, B. Shiotani, and R. Thompson, "Developing and distributing a CubeSat model-based system engineering (MBSE) reference model," in *Proceedings of the 31st Space Symposium*, pp. 1–14, 2015.

[11] P. Hammarström and E. Herzog, "Experience from integrating domain driven software system design into a systems engineering organization," *INCOSE International Symposium*, vol. 26, no. 1, pp. 1192–1203, 2016.

[12] R. Malone, B. Friedland, J. Herrold, and D. Fogarty, "Insights from large scale model based systems engineering at Boeing," *INCOSE International Symposium*, vol. 26, no. 1, pp. 542–555, 2016.

[13] S. Friedenthal and R. Burkhart, "Evolving SysML and the system modeling environment to support MBSE," *INSIGHT*, vol. 18, no. 2, pp. 39–41, 2015.

[14] A. L. Graham Bleakley and A. Whitfield, "Determining the right solution using SysML and model based systems engineering (MBSE) for trade studies," *INCOSE International Symposium*, vol. 21, no. 1, pp. 783–795, 2011.

[15] P. Roques, "MBSE with the ARCADIA method and the Capella tool," in *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, 2016.

[16] C. A. Ericson, "Fault tree analysis," *Hazard analysis techniques for system safety*, pp. 183–221, 2005.

[17] D. H. Stamatis, *Failure mode and effect analysis: FMEA from theory to execution*. ASQ Quality Press, 2003.

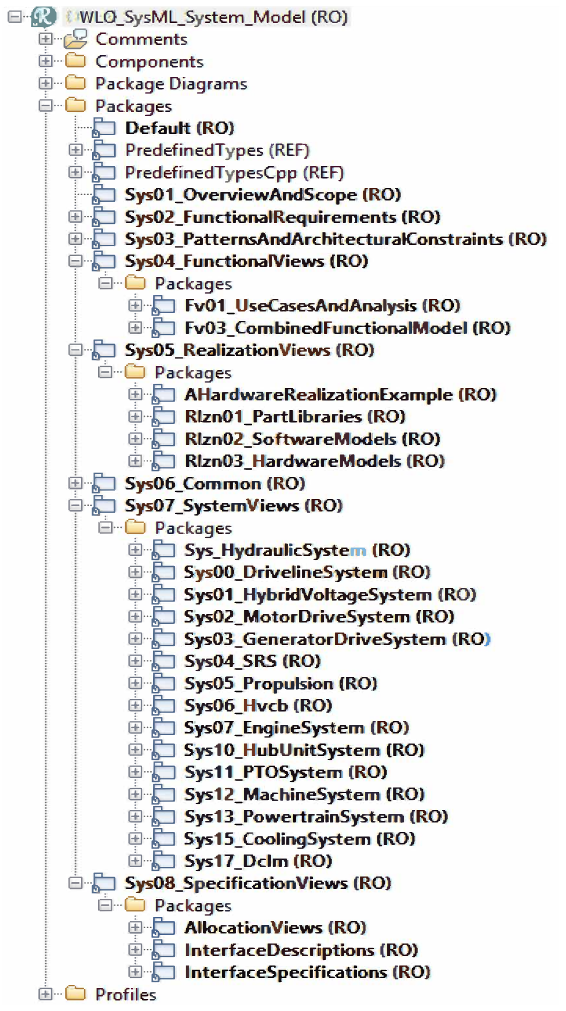
[18] S. Bonnet, J.-L. Voirin, V. Normand, and D. Exertier, "Implementing the MBSE cultural change: Organization, coaching and lessons learned," *INCOSE International Symposium*, vol. 25, no. 1, pp. 508–523, 2015.

APPENDIX

The modeling has been done using IBM Rational Rhapsody tool. The snapshot of the overview of the SysML System Model for the Drive Line System (DLS) is shown in Fig. 7.

In order to perform braking (the example behavior described), the following subsystems and entities are defined. Fig. 8 presents the BDD (Block Definition Diagram) for the logical view of the corresponding product break-down structure.

- **Operator** represents a person who operates the machine by sending a "braking request" through appropriate interface.
- **World** represents the machine environment aspects such as ground, temperature etc.



```
WLO_SysML_System_Model (RO)
    Comments
    Components
    Package Diagrams
    Packages
        Default (RO)
        PredefinedTypes (REF)
        PredefinedTypesCpp (REF)
        Sys01_OverviewAndScope (RO)
        Sys02_FunctionalRequirements (RO)
        Sys03_PatternsAndArchitecturalConstraints (RO)
        Sys04_FunctionalViews (RO)
            Packages
                Fv01_UseCasesAndAnalysis (RO)
                Fv03_CombinedFunctionalModel (RO)
        Sys05_RealizationViews (RO)
            Packages
                AHardwareRealizationExample (RO)
                Rlzn01_PartLibraries (RO)
                Rlzn02_SoftwareModels (RO)
                Rlzn03_HardwareModels (RO)
        Sys06_Common (RO)
        Sys07_SystemViews (RO)
            Packages
                Sys_HydraulicSystem (RO)
                Sys00_DrivelineSystem (RO)
                Sys01_HybridVoltageSystem (RO)
                Sys02_MotorDriveSystem (RO)
                Sys03_GeneratorDriveSystem (RO)
                Sys04_SRS (RO)
                Sys05_Propulsion (RO)
                Sys06_Hvcb (RO)
                Sys07_EngineSystem (RO)
                Sys10_HubUnitSystem (RO)
                Sys11_PTOSystem (RO)
                Sys12_MachineSystem (RO)
                Sys13_PowertrainSystem (RO)
                Sys15_CoolingSystem (RO)
                Sys17_Dclm (RO)
        Sys08_SpecificationViews (RO)
            Packages
                AllocationViews (RO)
                InterfaceDescriptions (RO)
                InterfaceSpecifications (RO)
    Profiles
```

Fig. 7. The System Model: Package View

- **WLO** represents the complete machine, consists of two subsystems (w.r.t the braking behavior): *Requester* and *Powertrain*.

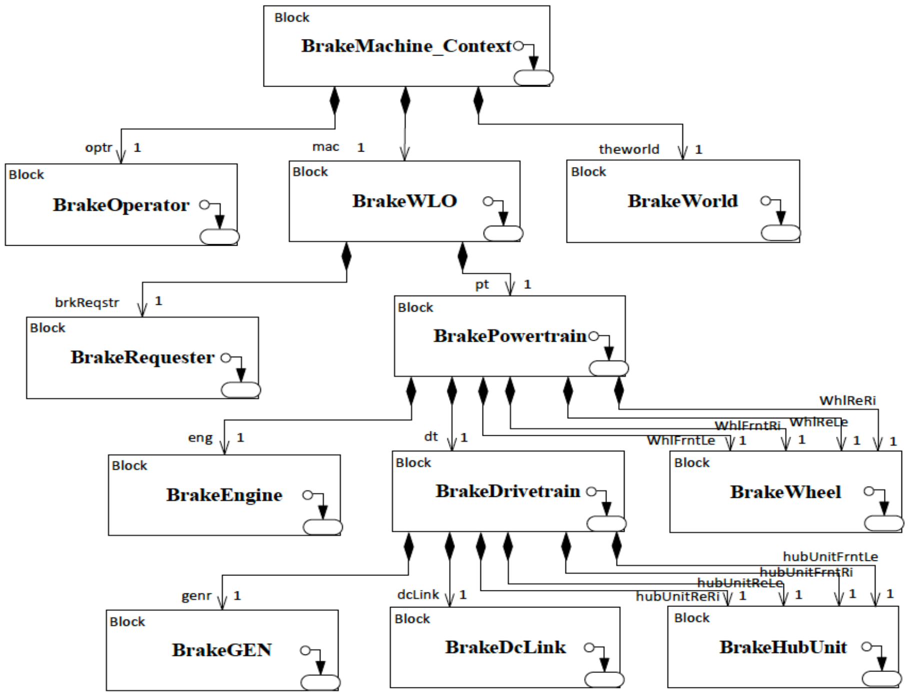


Fig. 8. Product Break-down Structure (logical)

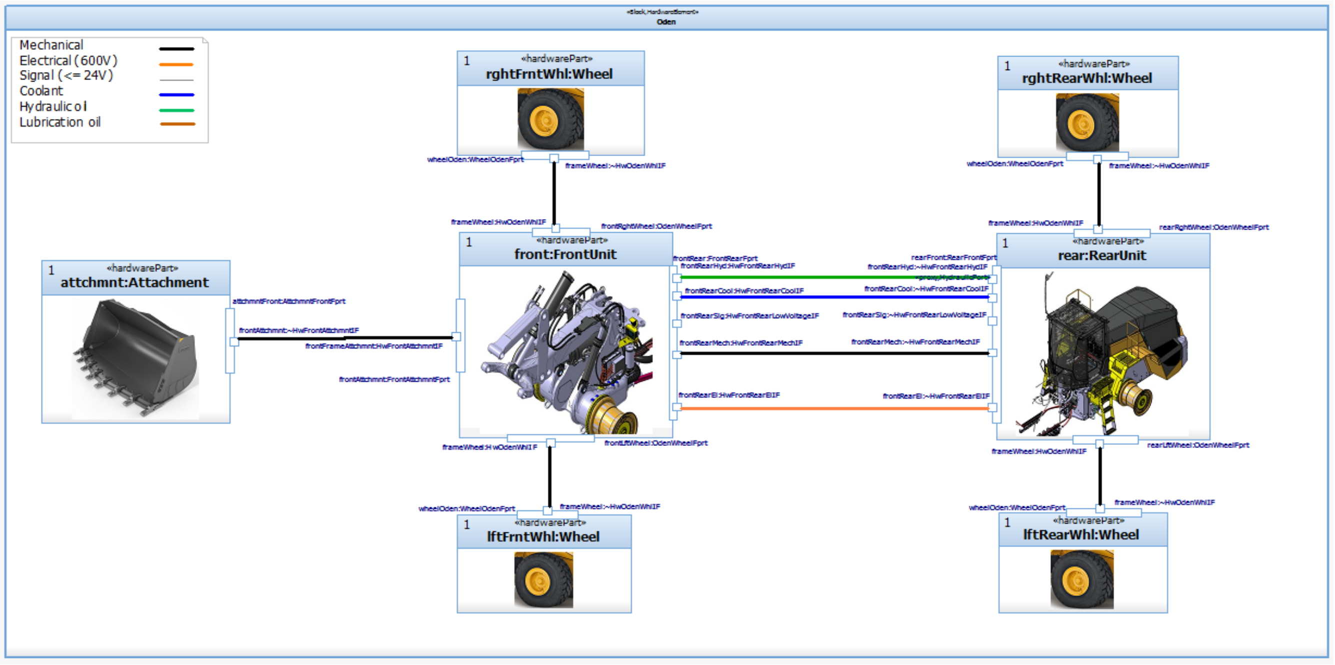


Fig. 9. Realization View

- **Powertrain** is a subsystem that includes following subsystems: *Engine*, *Drivetrain* and *Wheels* (#4).

Fig. 9 shows a partial "realization view" of the whole machine representing both hardware and software interfaces.