prostep IVIP

Recommendation

Recommendation SysML WF/IF

prostep ivip Recommendation 2023 PSI 28

SysML WF/IF

Version 1.0

Abstract

Collaboration based on SysML gets more and more important during the product development. On the one hand the importance of collaboration for system development is increasing and on the other hand the usage of SysML as standard for Model Based Systems Engineering (MBSE) is already considered in many companies.

In case of system modeling, the fact of having a common language does not mean at the same time, that there is a common exchange format available. Looking to current system development projects it is obvious, that collaboration scenarios where different system modeling tools are used, are not unusual. On top collaboration partners frequently use individual SysML profiles in combination with individual methodologies.

All the points make it a challenging situation for many companies to exchange system models and were reasons, why the SysML Workflow Forum (SysML WF) was founded in 2017. Besides the exchange of system models of course, related artefacts like requirements or test cases need also to be exchanged and bring the complexity of the exchange to an even higher level. The SysML WF decided to define three main use cases that were addressed in different work packages, to consider the main challenges of model exchange and the related artefacts. The three use cases are:

- Model Exchange
- Requirements Engineering & System Design
- Verification & Validation

In 2021 the SysML Implementor Forum (SysML IF) started its work, to have a closer look on the technical realization of the addressed use cases. Several vendors are participating inside the SysML IF representing different tool categories like authoring tools or traceability solution providers. Goal of the SysML IF is on the one hand to demonstrate current capabilities and approaches to the SysML WF and on the other hand to find technical solutions for the use cases and requirements addressed by the SysML WF.

This recommendation gives an overview on the findings and work results of both working groups, that have been elaborated so far. The different use cases are presented and an overview on the existing approaches for model exchange and existing exchange formats is given. The document closes with an outlook on the open points and the next steps.

Table of Contents

Table of Contents

Figures

Figures

Figures

Tables

Abbreviations, Definitions, References

Abbreviation	Meaning
API	Application Programming Interface
ASE	Advanced Systems Engineering
bdd	Block Defininition diagram
CAD	Computer-Aided Design
CDLC	Cross-Discipline Lifecycle Collaboration
CPO	Code of Openness (open initiative of prostep ivip)
CWM	Common Warehouse Metamodel
DDP	Digital Data Package (working group in prostep ivip)
EMOF	Essential Meta-Object Facility (OMG Standard)
FMEA	Failure Mode and Effects Analysis
FMI	Functional Mock-Up Interface (Open-source standard)
FMU	Functional Mock-Up Unit
FTA	Failure Tree Analysis
GfSE	Gesellschaft für Systems Engineering (German chapter of INCOSE)
ibd	Internal block diagram
IF	Implementor Forum (groups in prostep iViP)
(e)HSUV	(Extended) Hybrid Sports Utility Vehicle
INCOSE	International Council on Systems Engineering
IoT	Internet of Things
IREB	International Requirements Engineering Board
IP	Intellectual Property
JT	Jupiter Tesselation
MBSE	Model-Based Systems Engineering
MDK	Model Development Kit
MMS	Model Management System
MOF	Meta-Object Facility (OMG Standard)
OEM	Original Equipment Manufacturer
OMG	Object Management Group

Abbreviation	Meaning
OpenMBEE	Open Model-Based Engineering Environment
OSLC	Open Services for Lifecycle Collaboration
PDM	Product Data Management
PLM	Product Lifecycle Management
ReqIF	Requirements Interchange Format (OMG Standard)
RFI	Request for Information
RFP	Request for Proposal
RQM	Requirements Management
SE	Systems Engineering
SmartSE	Smart Systems Engineering (working group in prostep iViP)
SOI	System of Interest
SoS	Systems of Systems
SSP	System Structure and Parameterization (Modelica Standard)
SpecIF	Specification Integration Facility (Open-source standard of GfSE)
SVG	Scaleable Vector Graphics (W3C Standard)
SysML	System Modeling Language (OMG Standard)
UML	Unified Modeling Language (OMG Standard)
UUID	Universally Unique Identifier
VDA	German Association of the Automotive Industry
VDI	German Association of Engineers
V&V	Verification & Validation
WF	Workflow Forum (working group in prostep ivip)
WLTP	Worldwide Harmonized Light-Duty Vehicle Test Procedure
WPx	Work Package x
XMI	XML Metadata Interchange (OMG Standard)
XMI-DI	XMI Diagram Interchange
XML	Extensible Markup Language (W3C Standard)
XSL	Extensible Stylesheet Language (W3C Standard)

1 Management Summary

Several parallel activities are dealing with SysML today: Within prostep ivip e.V., the SmartSE project, which is dealing with SysML model exchange in collaborations between external partners. And also the CDLC-Forum, its successor the ICF project group or the DDP project group are focusing on internal cross-domain collaboration and linked data approaches.

Within OMG, SysML Version 2.0 is currently being developed, partly based on input provided by GfSE/INCOSE. To align the projects which are dealing with SysML within prostep ivip, to form one voice towards other organizations, and to enable industry to bring in their requirements, the prostep ivip SysML Workflow Forum (SysML WF) started in 2017.

The SysML Workflow Forum's vision is to establish industrial-level collaboration using Model-Based Systems Engineering to support handling of complexity. Therefore, the elaboration of consolidated best practices and analysis of limitations of MBSE based on industrial use cases is the key mission of the SysML WF.

Based on practical use cases it is examined for what purposes SysML is used in the industry and to what extent today's solutions and the modelling language support the needs of the users themselves, and what has to be improved for a widespread industrial application of SysML in the future. As results of this work, best practices focusing on OEM-Supplier collaboration are published as well as needs for action are elaborated and consolidated.

The focus was on the possibilities of SysML model exchange based on industrial collaboration use cases, verified by execution of derived test scenarios. In addition, two further work packages, "Systematic V&V management based on system models" and "Requirements exchange between OEM and supplier incl. traceability to affected system architecture artifacts" were defined, addressing the important topic of integrating system architecture models with related artefacts.

Driven by the SysML WF outcome and demand, the SysML Implementor Forum (SysML IF) has been setup in 2020 as new project group of prostep ivip e.V., which aims to establish technologies for tool interoperability based on SysML in industry. The SysML IF is a neutral forum for software vendors where they can test the applications that they are developing in an atmosphere of mutual trust and exchange information on experience already gained. SysML IF members presented their functionality and capabilities based on user requirements. They came to an agreement on common approaches to increase users' confidence in SysML including data exchange and collaboration by providing interoperable software.

2 General Aspects of Systems Engineering and SysML

A system is an unambiguously defined unit which consists of various subsystems and system elements (hardware, software, services, people, etc.) as well as their relations. The system element represents the lowest system hierarchy level. The System of Interest has a system boundary that is to be defined by the developer considering the respective perspective and interacts with Related Systems and its environment.

Figure 1 displays System Elements, System of Interest, Related Systems and its environment according to INCOSE SE Handbook.

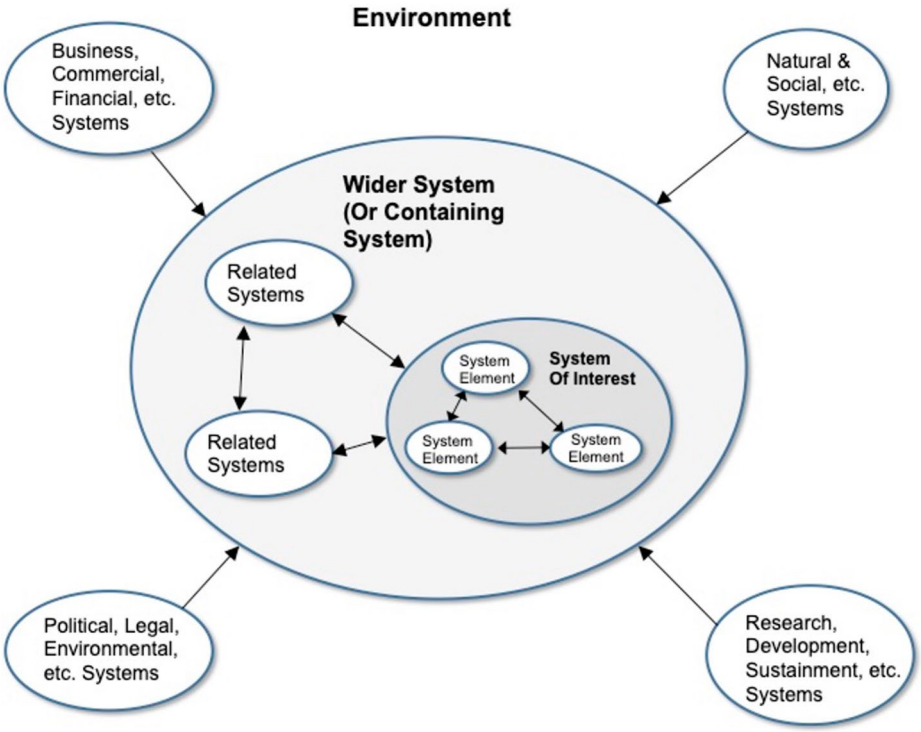


Figure 1: System Environment (according to INCOSE SE Handbook)

A higher-level system consists of several independent subsystems whose interaction contributes to the fulfilment of a required overall function. With a rising complexity of higher-level systems ranging from mechatronic to cyber-physical systems (CPS) and a trend from single products to System of Systems (SoS) where the overall system consists of multiple subsystems and the integration of multiple domains is a mandatory step, new approaches for development are demanded. One of these approaches is Systems Engineering (SE):

"Systems Engineering is a transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods." (INCOSE: Systems Engineering Definition, 2019)

The current Systems Engineering standard "ISO/IEC/IEEE 15288: 2015 – Systems and software engineering – System Life Cycle Processes" establishes a common framework of process descriptions for describing the life cycle of systems created by humans. It defines a set of processes and associated terminology from an engineering viewpoint. These processes can be applied at any level in the hierarchy of a system's structure. Selected sets of these processes can be applied throughout the life cycle for managing and performing the stages of a system's life cycle. This is accomplished through the involvement of all stakeholders, with the ultimate goal of achieving customer satisfaction (see Figure 2).

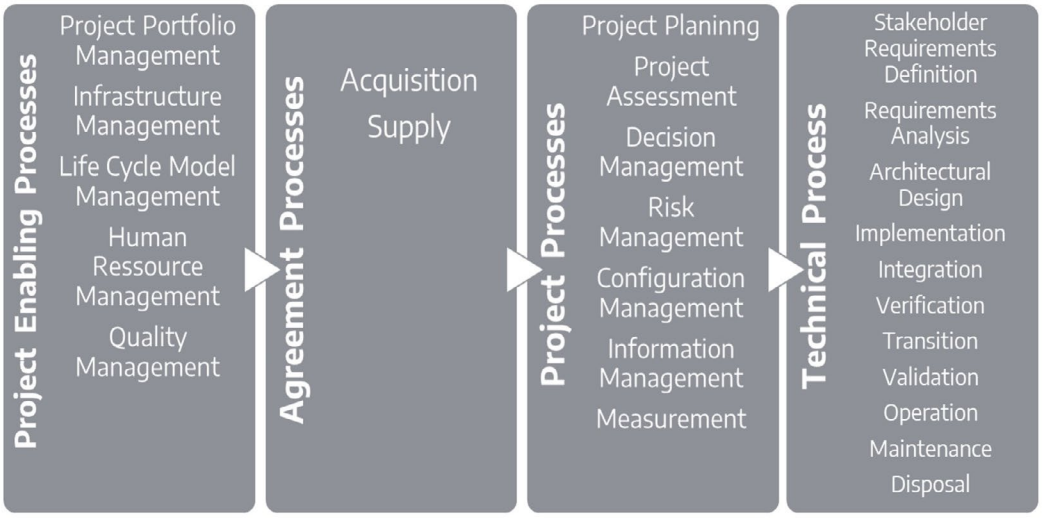


Figure 2: Systems Engineering Processes according to INCOSE SE Handbook (:em)

Engineering today is characterized by close interaction between the disciplines of mechanical, electrical and software engineering. A systematic and interdisciplinary approach is indispensable for the development of mechatronic and cyber-physical systems (CPS). For this purpose, the standard VDI 2206 was initially published in 2004. Due to the increasing networking with the IoT and Services as well as the high interdisciplinarity, complexity, and heterogeneity of the systems, a revision of this standard has become necessary and was published in 2021.

The holistic view of a system, the so-called "systems thinking", represents a core element of the V-Model according to Figure 3. The V-Model describes an inherent factual logical connection of tasks such as requirements elicitation, system architecture design and implementation of system elements, integration, verification and validation as well as transition in interdisciplinary product development. It does not represent a process model, but rather serves as a framework that describes the connection of tasks in the development of mechatronic and cyber-physical systems.

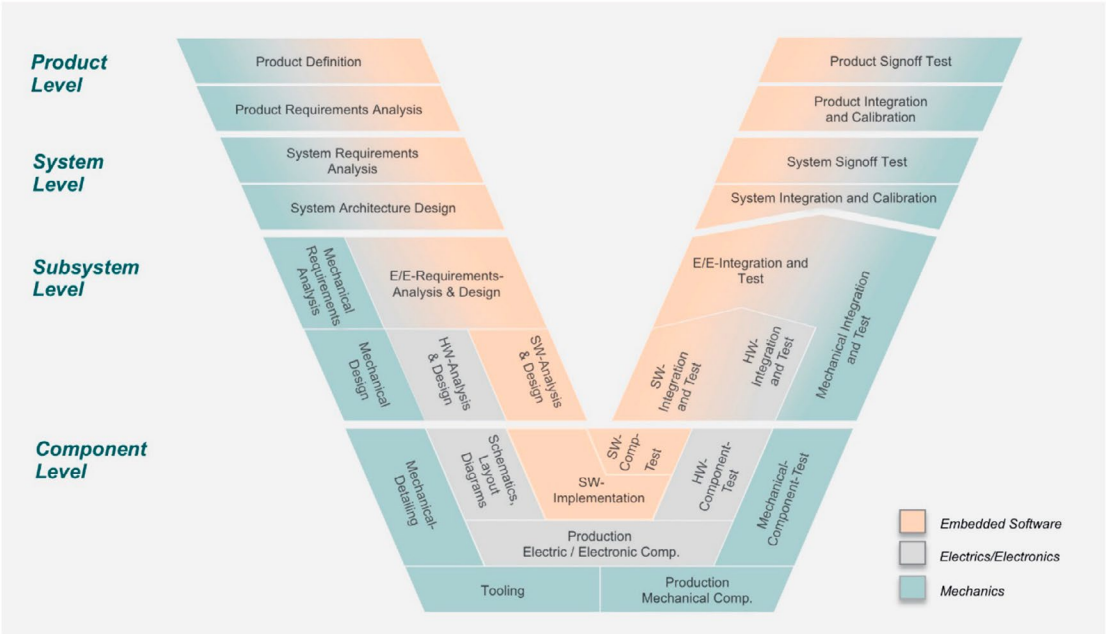


Figure 3: V-Model according to Smart Systems Engineering Recommendation V2

All requirements from upstream and down-stream product life-cycle phases shall be considered in accordance with systems engineering to do justice to the extensive technical and organizational changes.

The classical disciplines of mechanics, electrics/electronics and software engineering use their own terminologies, methods specific knowledge as well as CAx applications. The resulting limited perspectives and knowledge make inter-disciplinary collaboration difficult, but are needed for a holistic systems engineering approach. The V-Model starts there and describes a holistic methodological support. One of the essential special features of the V-Model is the dedicated break-down of the complex task into functions, logical and physical/technical structure (so called F-L-P/T approach) including partitioned subsystems and system elements (so called decomposition). The system elements implemented in the disciplines are gradually integrated again into subsystems and an overall system (integration). System properties are continuously verified and validated (so called V&V process) during this process.

2.1 Model-based Systems Engineering (MBSE)

The classical SE process was document-based and therefore hard to keep updated and understandable for every domain. The interdependencies of the documents are hard to track and the revision has to be done manually for every change of a document. Hence, a digital representation of the system is required. Modelling and analysis tasks are key and Model-based Systems Engineering (MBSE) has been introduced.

"Model-based Systems Engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases." (INCOSE: Systems Engineering Definition, 2019)

A model is a digital representation of the system in which its structure, interfaces and functions are shown in a simplified way. Models are created for overall systems, subsystems and system elements and differ in the problem considered and the level of detail. The usage of views on the modeled data, supports the overall and domain-specific understanding. Additionally, the system model can be seen as single-source-of-truth, which is constantly updated and thus eliminates manual effort to maintain different documents. With the usage of trace links, which are modeled dependencies between models or elements of models, the influence of changes and impacts can be analyzed and tracked down to every artefact in the development process.

In the context of Model-based Systems Engineering (MBSE), the models of the disciplines are linked to the models at the system level, enabling a consistent, redundancy-free exchange of information. For MBSE the modelling language SysML is one of the most used languages and supported by different modelling methods and tools. The application of MBSE serves to change from document-based to model-based development, with the effect of acceleration within the product development phase.

Kleiner et al. have presented an overview of possible use cases in MBSE which is shown in the following Figure 4.

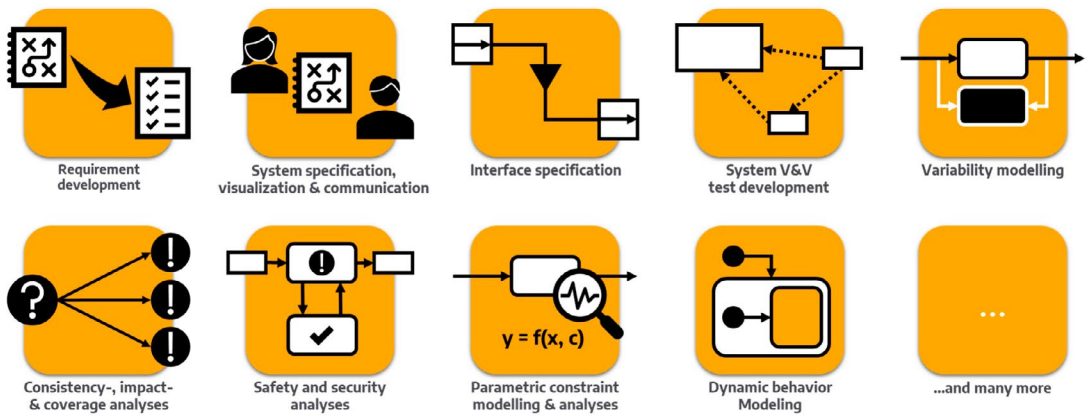


Figure 4: Use cases for MBSE (Kleiner, Lindemann, Korobov, & Hamester, 2018)

The use cases shown in Figure 4, including Documentation & Specification, Communication, Interface Definition, Dynamic Behavior Modeling, Impact Analysis, Derivation of Test Cases, Risk Analysis and Source Coding rely on an overall system model seen in the middle of this figure. This system model "[…] includes system specification, design, analysis, and verification information. It consists of model elements that represent requirements, design, test cases, design rationale, and their interrelationships." (Friedenthal, Moore, & Steiner, 2011)

Many companies see Model-based Systems Engineering (MBSE) as decisive for the efficient implementation of Systems Engineering. It is also seen as a foundation for the realization of comprehensive digital continuity. The system models should serve in the future as the central source of essential development artefacts for requirements, architecture, testing and realize a holistic support along the development phase. MBSE is the prerequisite for the continuity of development work and is thus the key concept for the success of Systems Engineering. However, a series of deficits such as the lack of an amortization concept for the model description is hampering the rapid spread of MBSE (Dumitrescu, Albers, Riedel, Stark, & Gausemeier, 2021).

2.2 System Modeling Language (SysML)

In recent years, the System Modeling Language (SysML) has been widely used for model-based system definition and architecture design. It can be used for the specification of requirements, the definition of system structure, functional architecture, system architecture and (partly) behavior modeling. Figure 5 gives a schematic view on the four major pillars of SysML.

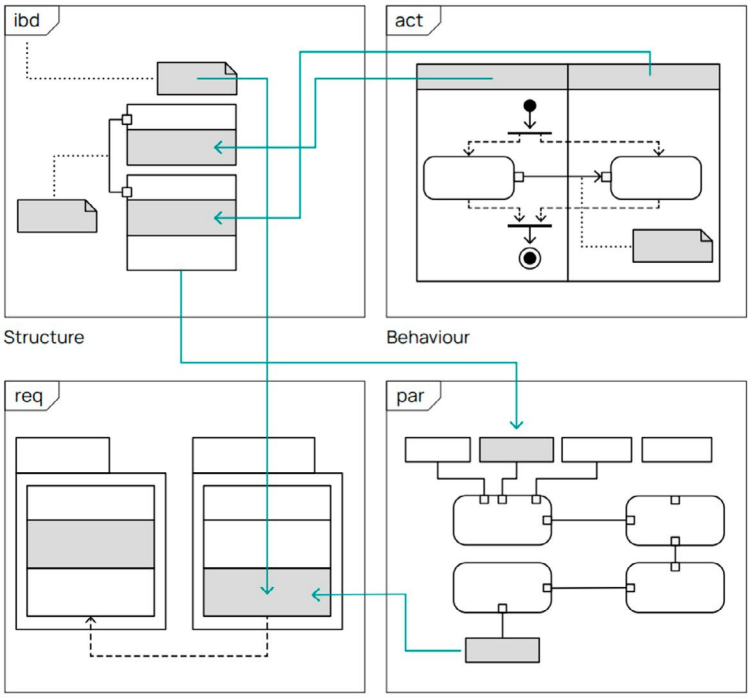


Figure 5: SysML aspects and the four major pillars (Dumitrescu, Albers, Riedel, Stark, & Gausemeier, 2021)

SysML has been and still is actively developed by the Object Management Group (OMG). The specification is available since November 2019 as version 1.6 (Object Management Group, 2019). Version 1.7 and 2 are currently under development in parallel. The current version 1.6 offers the following types of diagrams for the representation of model elements:

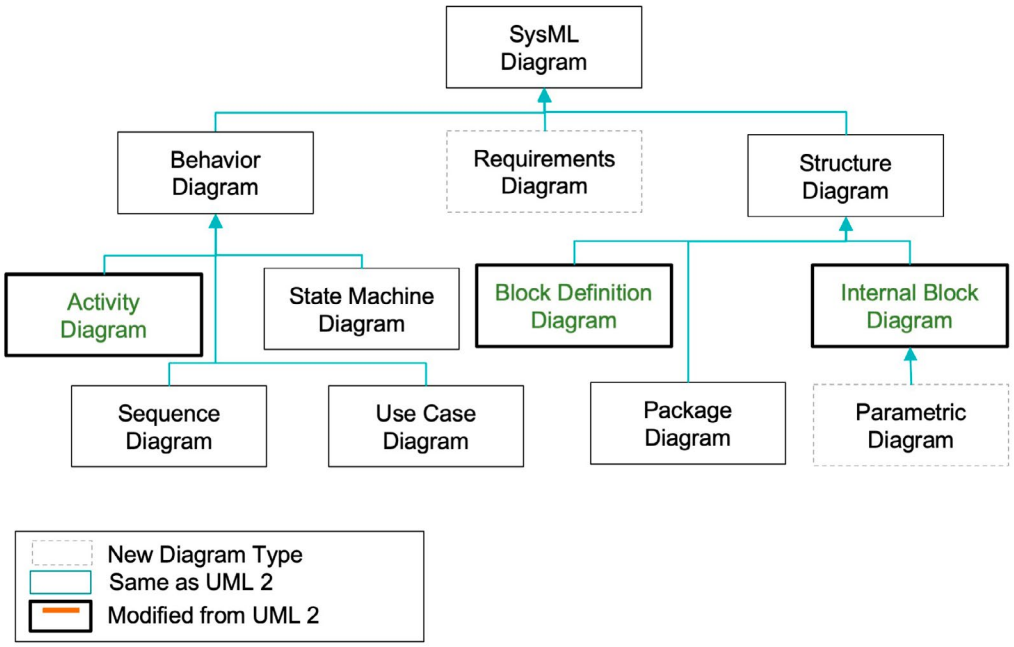


Figure 6: SysML Diagram types (Object Management Group, 2019, S. S. 211)

Beside these diagram types, there is a variety of so-called stereotypes, attributes and profiles, which can be used to extend the functionality of SysML and also for the personal customization. An example is the Safety and Reliability Analysis, which defines new stereotypes and profiles to include Safety aspects into UML (Unified Modeling Language), which is currently the base for SysML v1.x. With these profiles, UML is extended with the functionality to define safety and functionality aspects like Fault Tree Analysis (FTA) and Failure Mode and Effects Analysis (FMEA). Many companies define their own profiles and stereotypes to adapt SysML to their needs or their own development method requirements.

The second version of SysML, known as SysML v2, is meant to be the next step of SysML to support graphical and textual modeling in MBSE. The SysML v2 will be based on his own Metamodel Kernel Modeling Language (KernML) and will also include an API for accessing SysML models. Weilkiens described it as the "next generation modeling language for the next 15-20 years" (Weilkiens, 2019).

With the final version of SysML v1.0 in 2007, a Request for Information (RFI) has been started in 2009 followed by a Request for Proposal (RFP) Development from 2014 until December 2017. Since then, the workgroup SysML Submission Team (SST) is collecting feedback to the RFP for the further development of SysML v2. The submission, which has been shifted from 04th of November 2019 to Q2 2020, is set as starting point for the development of this SysML version. A final version might be seen in 2023. Figure 7 shows the current Roadmap of SysML v1 and v2.

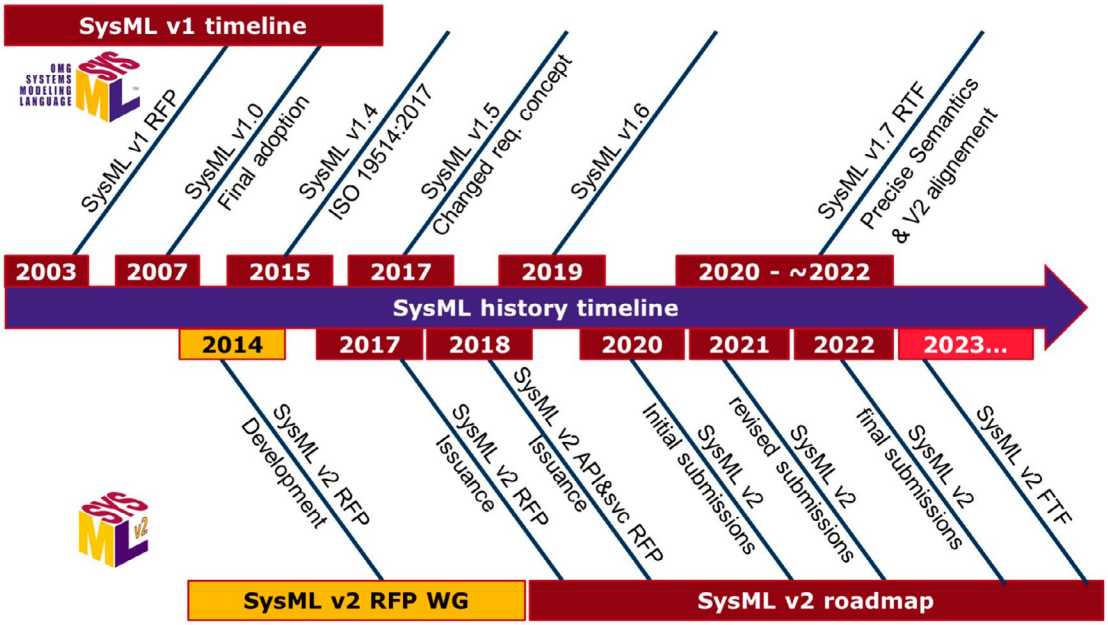


Figure 7 SysML v1 timeline and v2 roadmap taken from Uwe Kaufmann, GfSE AG PLM4MBSE

Weilkiens has presented some features of the upcoming SysML v2 – the next generation modelling language – including the following (Weilkiens, 2019):

- New language architecture: The new version will no longer be based solely on UML and thus be not restricted by the information-based UML. A new language architecture called Kernel Modeling Language (KernML) is in development shall be used as basis for SysML v2. To allow an easier adoption SysML v2 profiles for the current UML based standard are developed in parallel. This architecture is the base for the further features.
- Version and Timestamp: As versioning and timestamping is important in today's development procedures, SysML v2 shall offer a possibility to apply timestamp and version data directly onto the model elements and not only to the full model or diagrams.
- Data protection: Data protection control for model elements shall be implemented in SysML v2 as well.
- Cause-Effect-relationship and Risk: As has been shown by the example of Biggs et al. on top, the Safety and reliability aspects are currently not consistently included and shall be included directly in SysML v2.
- Navigation: SysML v2 shall include a hyperlinking functionality to link model elements internal or externally.
- Variant Modeling: The ISO 26550 for product line engineering and management shall be fulfilled by variant management integrated in SysML v2. This includes variation points, variants, variability expressions and variant binding.
- New types: Beside the primitive types from SysML v1, informatics types like integers, reals, strings, booleans, times/dates and complex there shall be an implementation of collection types like sequence, set, ordered set, bag, vectors, matrices and higher order tensors. Thus, it is possible to describe most of the imaginable mathematical & physical problems as well.
- Materials: As an important step towards interaction with 3D representations, SysML v2 shall implement named material and property representation in model libraries and assignment to different physical components.
- Usage-oriented modeling: In the current version of SysML it is necessary to define block definition diagrams (bdd) as a framework before the actual interrelationship can be modeled in internal block diagrams (ibd). SysML v2 offers the possibility to directly model the ibd's.
- Support of Digital Twins: It shall be possible to model Digital Twins. Currently there has been no further information, how this shall be achieved.
- Interfaces between domains: The interfaces between the domains E/E, Mechanics, Informatics, Logical Design and Requirement Design shall be easier to connect.
- Harmonization of sequence and activity diagram: The previous activity and sequence diagram have been doing the same but in different forms, without supporting each other. With the upcoming version, they shall be usable as different diagram representations of the same model elements.

- Behavior and structure: The integration between behavior and structure shall be enhanced considering mainly their inputs and outputs.
- Model-Based Requirements Engineering: While in current SysML versions the modeling of requirements is mainly the combination of blocks that are filled with text, the v2 shall bring a fully model-based approach, where textual requirements are only supported but not mandatory.
- Support of analysis and decisions
- Basic geometries: Basic 2D and 3D geometries, including their base coordinate frame, shall be representable by the SysML v2. This allows an even better visualization and understanding.
- New diagrams: Beside the classic diagrams, tables and matrices as well as text, dynamic visualization and the already mentioned geometry visualizations shall be included directly in SysML v2.
- Test Cases for conformance: To support the implementation of SysML v2 test cases for conformance checks shall be included in the meta model and profiles as well.
- API and services: Provide a standardized, tool-independent API and basic services in order to access a SysML model. The standard will make it possible to write applications using the API and services independent of a specific SysML too. In order to allow tool-independence and easier implementation, a standardized and tool-independent Application Programming Interface shall be included, that allows reading and writing access to the system model without the need for a specific SysML-tool

As described MBSE is not only to be understood as the usage of SysML. SysML is a standardized and widely spread graphical modeling language that allows an easier and understandable definition of System Models and the mentioned use cases of Figure 4. The prostep ivip SysML Workflow Forum (SysML WF) investigates how models described with SysML can be used for collaboration in different domains. Beside SysML there are multiple other languages and methods such as OPM or Arcadia that can be used for the system definition.

2.3 Additional Standards for Collaborative MBSE

Further standardized formats for the usage in MBSE have already been presented in the PSI for collaborative Systems Engineering based on IT Standards and can be seen in the following Table 1. In the early stage of development, the standards ReqIF (exchange of requirement models), XMI (XML Metadata Interchange) and FMI (Functional Mock-Up Interface for the coupling of simulation models) (FMI Standard, 2022) have attention in this domain, while JT as visualization format is mainly used in the late development phase. OSLC (Open Services for Lifecycle Collaboration) mentioned last in the following table is getting more attention as overall framework.

The SysML WF is open for each format that could help improving the exchange of models. Therefore, e.g. the XMI format was discussed as a possible useful standard for the model exchange. Some vendors like for example Dassault Systèmes (former NoMagic) use it as their native format for their system modeling tools.

Reference/Short name	Description/Content
Functional Mock-Up Interface (FMI)	Interface to combine multiple models, mainly used for co-simulations
ISO 10303-233 – Systems Engineering	Application protocol for the representation of system engineering data, independent from the domain
ISO 10303-242 – Application Protocol: Managed model-based 3D engineering	AP242 XML, used to store relevant information, that shall not be directly applied to the JT file
SysML v1.6	Current SysML standard
SysML v2	Upcoming SysML standard
XML Metadata Interchange (XMI)	XML based format to exchange metadata, expressed in Meta-Object Facility (MOF)
Open Services for Lifecycle Collaboration (OSLC)	XML based format to exchange metadata, expressed in Meta-Object Facility (MOF)

Table 1: Standards related to SysML and MBSE

2.4 Working Groups for MBSE

As MBSE is an important and broadly investigated topic, multiple working groups are focusing on MBSE related topics. In the following short overview, some of the most relevant working groups for the combination of SysML and MBSE shall be mentioned.

As top-level organizations the prostep ivip, the OMG and the International Council of Systems Engineering (INCOSE) with its German chapter "Gesellschaft für Systems Engineering" (GfSE) have been chosen. They shall be considered as partners for further investigations in the current topic.

Prostep ivip	OMG	INCOSE	GfSE	Others
• Digital Data Package (DDP) • Smart SE • SysML WF/IF • Integrated Collaboration Framework (ICF)	• CORBA • CWM (Common Warehouse Metamodel) • DDS • Model Driven Architecture • Meta-Object Facility • SysML • UML	• Complex Systems • Digital Engineering Information Exchange Working Group • MBSE Initiative • MBSE Patterns • System of Systems • Tools Integration & Model Lifecycle Management	• Formalization of Modeling (MF, German "Modellierung formalisieren") • Model based Systems Engineering (MBSE) • System Safety Modelling Language (Sys(S) ML) • PLM for MBSE (PLM4MBSE) • Specification Integration Facility (SpecIF)	• Open Model-based Engineering Environment (OpenMBEE) • Long Term Archving MBSE (LOTAR MBSE) • Dublin Core Metadata Initiative (DCMI)

Table 2: Working groups to consider for cooperation

2.5 Benefits, Challenges and Deployment of MBSE

This chapter gives an overview on benefits and challenges for the introduction of MBSE, as well as an overview on the current deployment in the industry.

Some major benefits of MBSE are:

- Handling of complex systems and SoS (System of Systems)
- Working on a common digital representation which is constantly updated
- Early validation and verification
- Easier communication between the different domains
- Enabling of interdisciplinary model-based collaboration

Challenges are mainly based on the understanding of systems as a whole instead of the currently widespread component-oriented view. Therefore, some major challenges are:

- Organizational change process
- Enable employees to think functional and more user-oriented instead physical and based on existing products (think in problem space, not in solution space)
- System-wide thinking and considering interfacing domains instead of focusing on the own domain

The current deployment is mainly set in the requirements, test and system architecture design phase on system/sub-system level and refinement of these requirements and interfaces in the individual sub-levels. Some companies have started approaches for an overall MBSE approach, some start implementing sub-aspects of the MBSE approach, such as the functional decomposition. Especially the automotive domain is considering the MBSE methods due to some regulatory affairs.

2.6 Advanced Systems Engineering (ASE)

Advanced Systems Engineering (ASE) is the guiding principle for the successful design of innovative products, services and product-service systems as well as their development process in future. The mission statement takes particular account of the effects of increasing digitization, interdisciplinarity and networking for mastering technical and organizational complexity in future engineering. ASE integrates system-oriented and highly innovative approaches to engineering and stands for a new perspective in the planning, development and operation of the technical systems of tomorrow. In December 2022 the Advanced Systems Engineering strategy was published (Albers, et al., 2022).

As a result of digital transformation, digitization is marking a continuous change from products via mechatronic systems to intelligent cyber-physical systems and solutions. These *Advanced Systems* hold enormous market potential, unique opportunities and considerable competitive advantages for pioneering businesses, the development of these systems requires new skills and qualifications of the people involved.

The collaborative, sustainable engineering of future products as well as of the product requires talented developers from a wide range of disciplines such as engineering, computer science, sociology and industrial science. In order to manage the complexity of this increasingly interdisciplinary development process, it is necessary to introduce and apply the skills, processes and methods of *Systems Engineering* across industry.

Advanced Engineering can surpass the current limits of engineering and revolutionize existing products and services. This includes, for example, the use of emerging technologies such as AI and the Digital Twin, as well as new work structures such as agility in engineering.

There is a particular potential for future value creation if the combination of *Advanced Systems, Systems Engineering* and *Advanced Engineering* work together. Based on the *Advanced Systems Engineering* approach and funded R&D projects, the players in economy and science will push the existing strengths and jointly pursue the goal of Germany as a location for innovation in the long term.

Based on the Product Life Cycle Management concept and the rise of MBSE, high heterogeneity has arisen among IT authoring tools and databases that need to be integrated. This uses a lot of personnel capacity not limited to engineering departments only. Hence, this situation must be counteracted by standards e.g. for data exchange formats to ensure data continuity and allow interfaces according to the Code of Openness (CPO), which is an open initiative of prostep ivip under the patronage of the German Federal Ministry for Economic Affairs and Energy (Germany).

3 Use Cases and Requirements for SysML IF

The Smart Systems Engineering (SmartSE) project group works together with a group of participants from almost 30 companies and research institutions to develop application-oriented concepts for overcoming common systems engineering challenges since 2012. To this end, SmartSE develops recommendations for process design, promotes technical standards for the cooperative development of complex mechatronic systems, and supports the creation of transparency with regard to systems engineering objects both within and across companies.

In 2017, the SysML WF was initiated and separated from SmartSE based on the rising demand towards SysML and specific use cases. The objective of prostep ivip's SysML WF project group is to identify the needs and requirements of industry with regard to both the Systems Modeling Language (SysML) itself and SysML tools. Based on the use case "Cross-enterprise model exchange throughout the entire development cycle (from requirements management to verification and validation)", which is particularly relevant to industry, the project group is examining to what extent today's solutions and the modeling language support the needs of the users and what has to be done to achieve full industrial applicability of SysML in the context of model exchange in the future. On the one hand, best practices for dealing with the challenges that system modeling currently poses are to be developed on the basis of the requirements identified. On the other hand, the working group aims to address the identified requirements relating to the SysML language and implementation of the corresponding tools in the respective committees.

In 2020, the SysML IF has been established according to other work groups for vendors, implementors and service providers.

Based on the input from SysML WF as well as its respective formats, different use cases and requirements have been presented, that are currently requested and need to be demonstrated by the system provider based on MBSE/SysML tools and be integrated or linked when it comes to tool interoperability.

The presented use cases are:

- Use Case 1 – Model Exchange
- Use Case 2 – Requirements Engineering & System Design (SysML – RQM)
- Use Case 3 – Verification/Validation (SysML – V&V)

The following Figure 8 gives an overview on the interaction between the different use cases and their context.

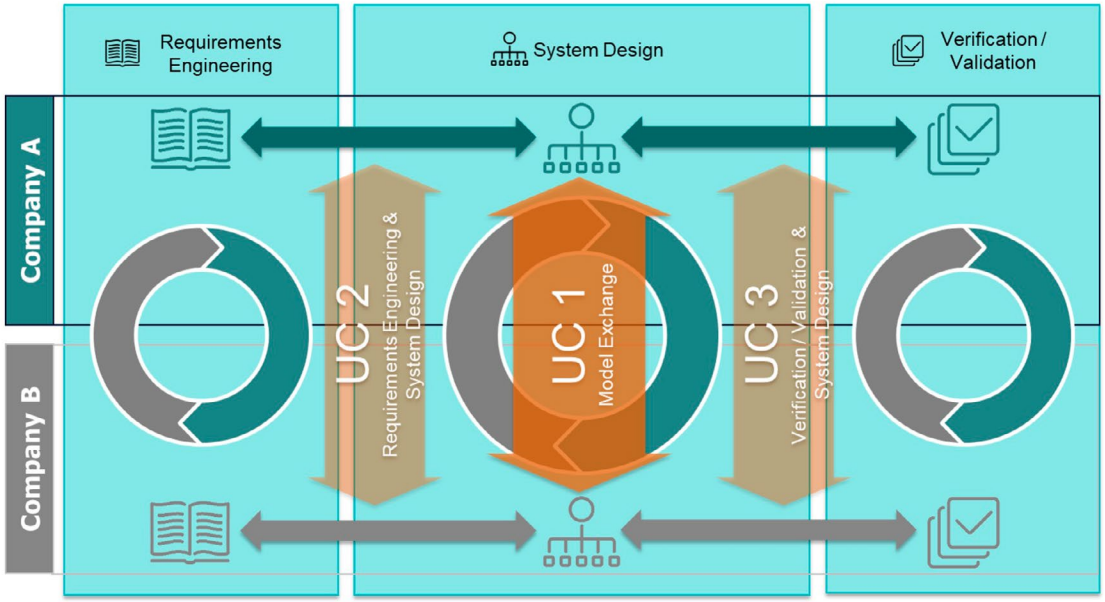


Figure 8: SysML WF Use Case overview

For each Use Case the SysML WF has a single working group, that discusses the needs and requirements that need to be addressed to the system provider participating inside the SysML IF. The following chapters describe the requirements and give an overview on the current status inside the SysML WF.

3.1 Use Case "Model Exchange"

The Use Case "Model Exchange" is discussed inside the Work Package 4 (WP4) of the SysML WF. The main use case that is considered inside this WP is the exchange of system models between two companies that use different system modeling tools. As an example, the collaboration between an OEM and a supplier is taken.

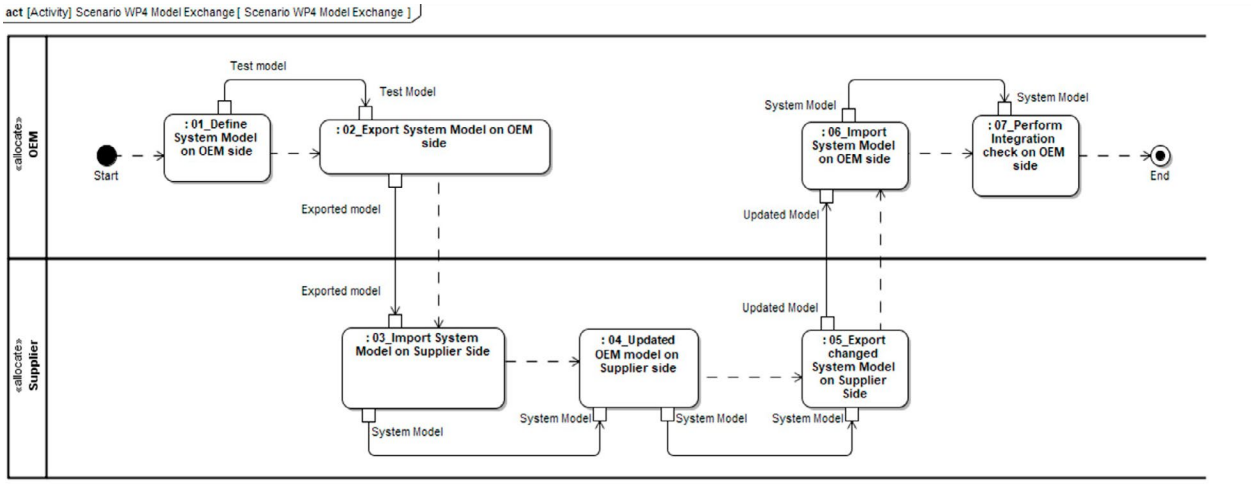


Figure 9: Collaboration scenario WP4 "Model Exchange"

The scenario describes a situation, that already occurs today and will get more and more relevant with the increasing importance of Model-based Systems Engineering. Companies are using system models for describing different aspects of their systems and products and specific information out of these models is also relevant for the exchange with their partners and suppliers. A possible scenario could be that the OEM provides the context of a sub-system towards the supplier and the supplier enriches the sub-system with a structure and behavior and returns it back to the OEM.

Like in other disciplines, an exchange is always difficult if there is no native interoperability between different tools that are used. For the exchange of geometric data between different CAD tools, several standard formats are available like JT or STEP and have been well established and improved during the last years.

For exchanging system models, such a standard is still missing. Despite the fact, that SysML is a standard, the implementation inside different system modeling tools is not consistent from a technical perspective, which makes an exchange of system models very difficult, depending on the specific exchange scenario. Also the fact, that in many cases partners use different SysML profiles and different modeling techniques for creating system models brings a special challenge on top.

For the use case of the model exchange, the SysML WF is open to every solution, that is offered by the different system providers. To check the capabilities of the different solutions, the SysML WF prepared a set of high-level requirements that summarize the key points, that must be fulfilled by a potential solution.

Name	Text
IP protection	In connection with IP protection, it shall be possible to define specific values, attributes, tags, etc. that should be exchange and not.
Linking of model elements	The parts of the model created by the OEM that are to be transferred to the supplier shall be linked to each other via relations.
Exchange of real model data	It shall be possible for the OEM to provide an "empty" model structure for the supplier, which is then specified and fully modeled by the supplier. (e.g. an ibd of a block with interfaces and expected behavior)
Exchange of (parts of) profile information	It shall be possible to exchange additional information to the model and its elements (semantic meaning is important in the exchange
Specification of requirements and expectations	The OEM shall enable the supplier to build up a system understanding by modeling the use cases, the requirements, the system context with interfaces to surrounding and the expected behavior.
Model exchange on different levels of abstraction	The model exchange between OEM and supplier shall be able to be repeated and performed on different levels of abstraction.
Exchange minimal set of SysML Diagrams and Elements	It shall be possible to exchange parts of a model to a partner and to reimport the result without loss of consistency.

Table 3: Overview High-Level Requirements WP4 "Model Exchange"

The use case considers, among others, these aspects:

- **IP protection:** System models contain a lot of information and give a deep insight into the dependencies inside the model. For this reason, there is the demand to control which information will be exchanged.
- **Exchange of diagrams:** A special challenge is the exchange of SysML diagrams. Even the diagrams are actually "only" a specific representation of the model content, the exchange of diagrams or at least the exchange of diagram visualizations would be helpful.
- **Exchange of link information:** System models provide important traceability information by linking different artefacts inside the system model. Therefore, the exchange of link information is an essential need for the exchange use case.

3.1.1 Details on Exchange Workflow

The generic overview of the engineering process and cooperation model between the OEM and supplier is shown in Figure 10. As it has been experienced that a "roundtrip" – meaning the export of a model to the supplier and then import back from supplier into the OEM's system – is a big challenge, the scope here is only focusing on exchanging a system model from the OEM to the supplier.

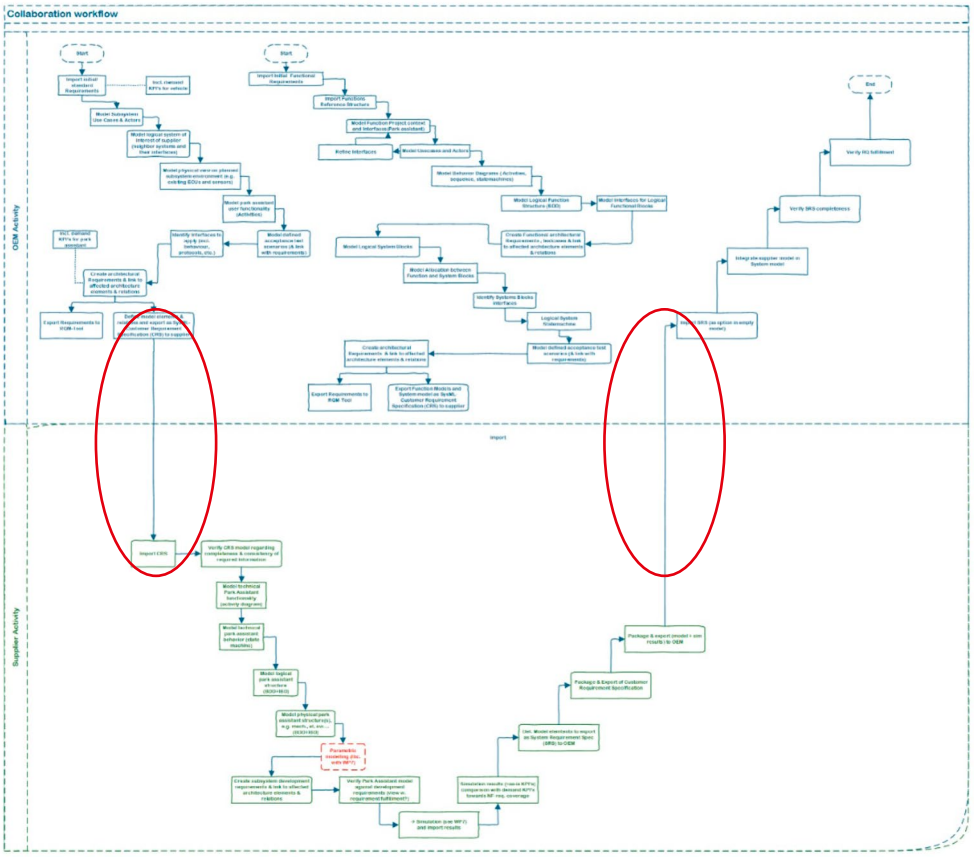


Figure 10: Detailed Exchange Workflow

As a first step the OEM defines the high-level system description and all information needed by the supplier to develop the sub-system. This covers a black box description of the required system (specification, no design), the structural context, the functional context and additional behavior diagrams.

Structural context:

The structural context is given by a black box specification of the system as well as by an external interface definition as shown in Figure 11.

The intention is to clearly define what is the system of interest (SOI), what are the neighboring systems, which constraints need to be considered and what environmental conditions shall be considered. Typically, this is done using an internal block diagram incl. the following modeling elements:

- Block/Part
- Interface Blocks
- Proxy Ports
- Connectors
- Flow Properties (in/out; direction)
- Comments
- Item Flows
- Associations

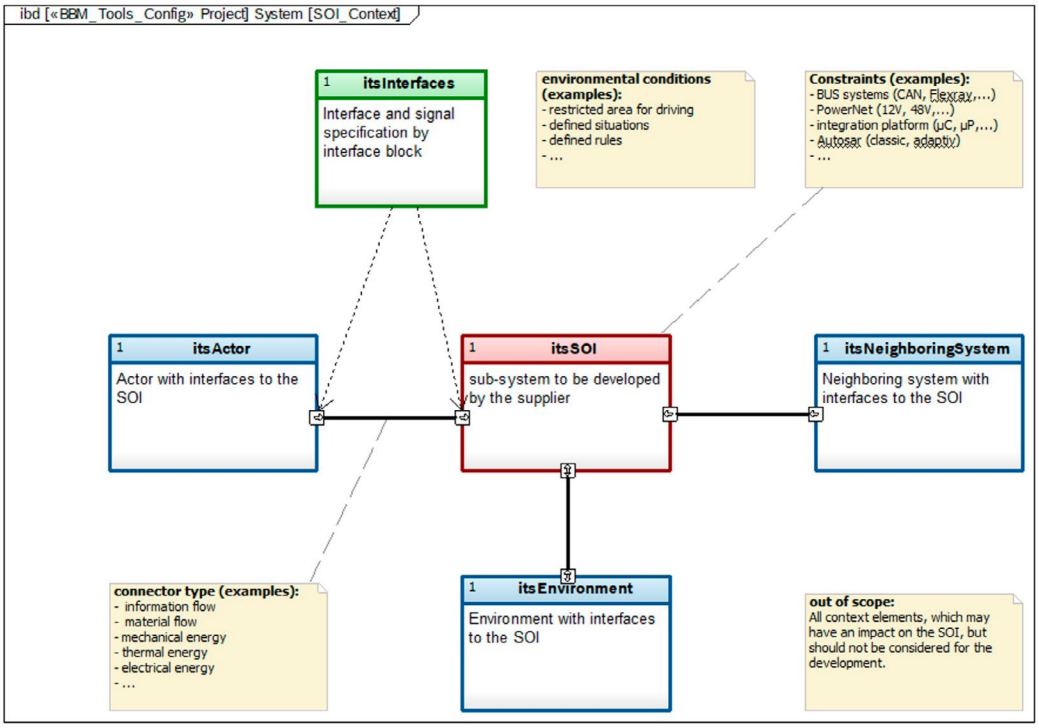


Figure 11: Structural Context

Functional context:
The functional context is given by a black box definition of the function describing what the system shall do as well as a definition of the external functional interfaces as shown in Figure 12.

The intention is to clearly define what is the user function, which function is supplied, which flows exist between context and functions and what is exchanged via these flows. Typically, this is done using an activity diagram incl. the following modeling elements:

- Activity + Parameter
- Types of Parameters
- Call Behavior Actions
- Action Pins (based on Parameters)
- Allocate Activity Partitions
- Object Flow
- Specific Element Properties
- Operations

In some cases, an internal block diagram is used instead of or in addition to the activity diagram.

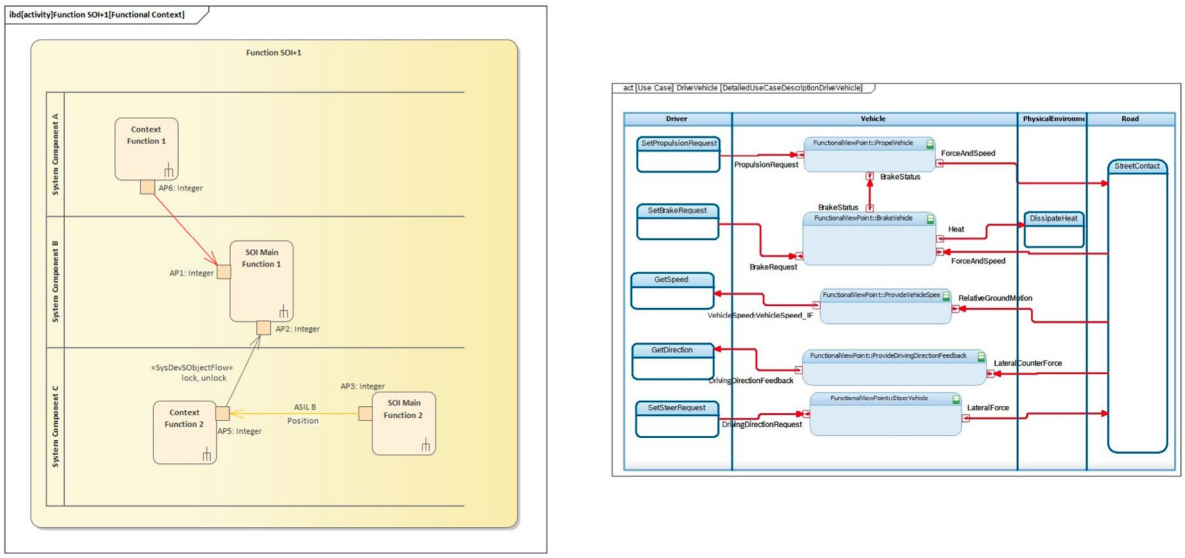


Figure 12: Functional Context

Further behavior diagrams:
Further behavior diagrams might be added to detail the intended black box behavior of the system, the required system states and the given constraints (timing etc.). Figure 13 shows as an example of a state diagram with transitions including trigger and actions.

The intention is to define clearly what is the expected/specified behavior of the system, what is the defined behavior of a function or component and which further constraints exist for system elements like timing or sequence of functions. Typically, this is done using a state machine diagram incl. the following modeling items:

- States
- Transitions
- Trigger & Guard Condition
- Blocks (as lifelines)
- Messages (operations)

In some cases, a sequence diagram is used instead of or in addition to the state diagram.

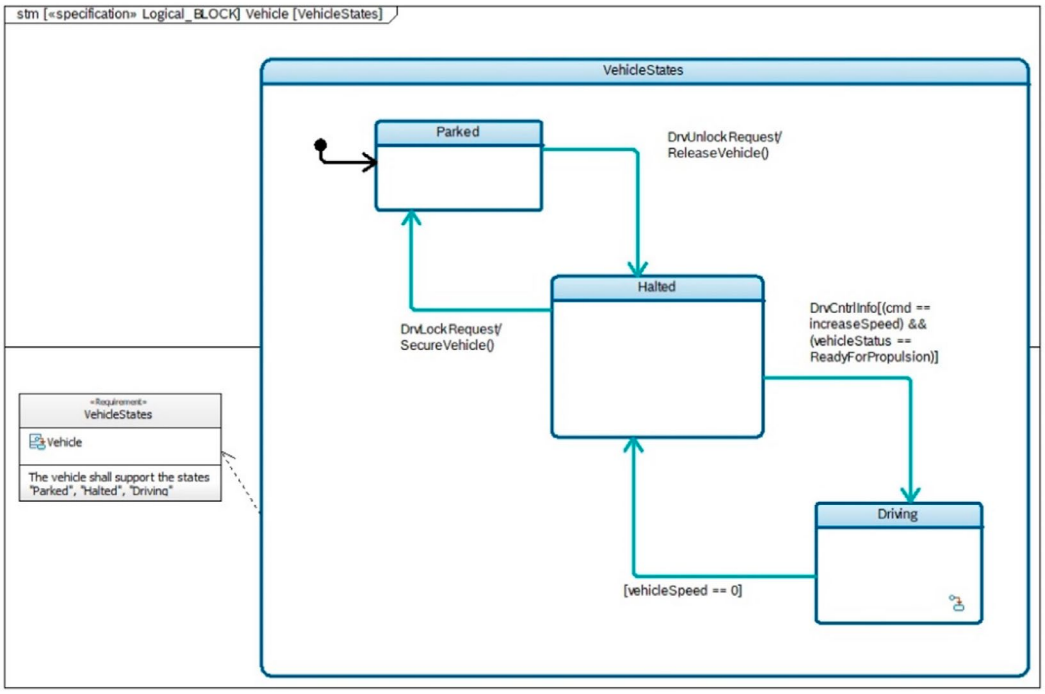


Figure 13: State Diagram with Transitions

3.1.2 Overview of Exchanged Model Content

Based on the defined diagrams that the OEM has to prepare for handover to the supplier, the minimum content of a system model that needs to be exchanged is defined in Figure 14. This refers to the SysML items that are most commonly used to specify systems with an MBSE approach and covers different kind of elements, connections between these elements, properties as well as different diagram types.

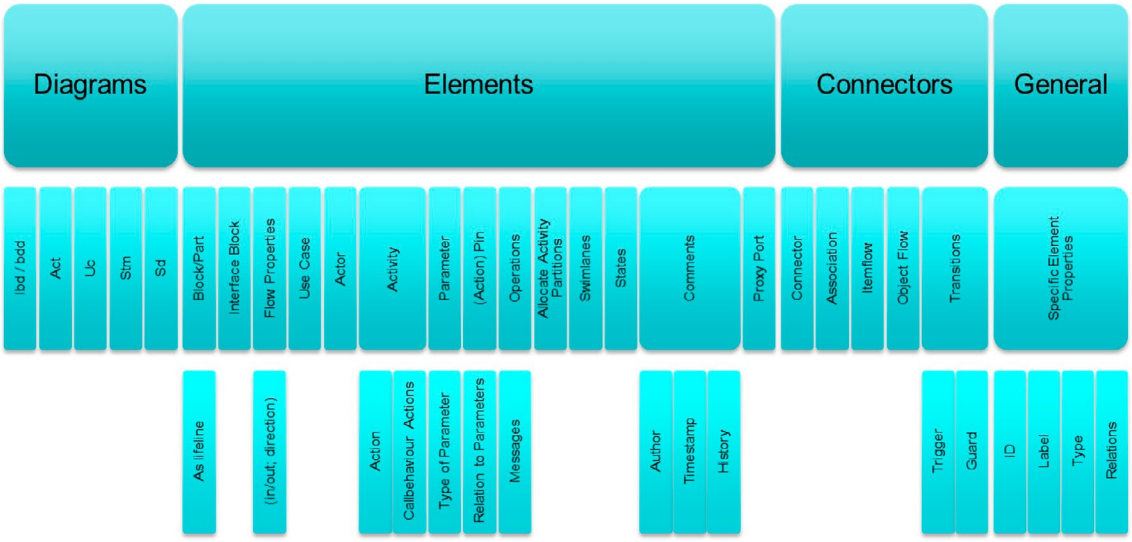


Figure 14: Overview of exchanged model content

3.1.3 IP Protection for Model Exchange

The considered system models should contain information about the system under development like its components, interfaces, or functions. But in addition to their intended purpose, models might also contain details about

- product variants not in scope of the current development job
- innovative neighboring systems or upcoming technologies
- product evolution plans or possible future applications
- internal data, like estimated costs
- internal development methods, like metamodels
- organizational structures up to names of business partners or individuals
- tool specific meta data, like templates, file paths, or timestamps of last modification

The more mature a model becomes, and the more individuals are contributing to the model, the harder it becomes to separate information to be shared with development partners across company borders from information to be kept internally. At this point, protection of intellectual property (IP) becomes a challenge when exchanging SysML models between companies.

Instead of exporting a model on the whole, a mechanism is needed to mark elements of a model as to be exported and other elements as to stay internally. Tools for model exchange shall take this into account and provide a possibility for IP protection.

A possible solution could be based on separate SysML packages for imported data, exported data and internal data:

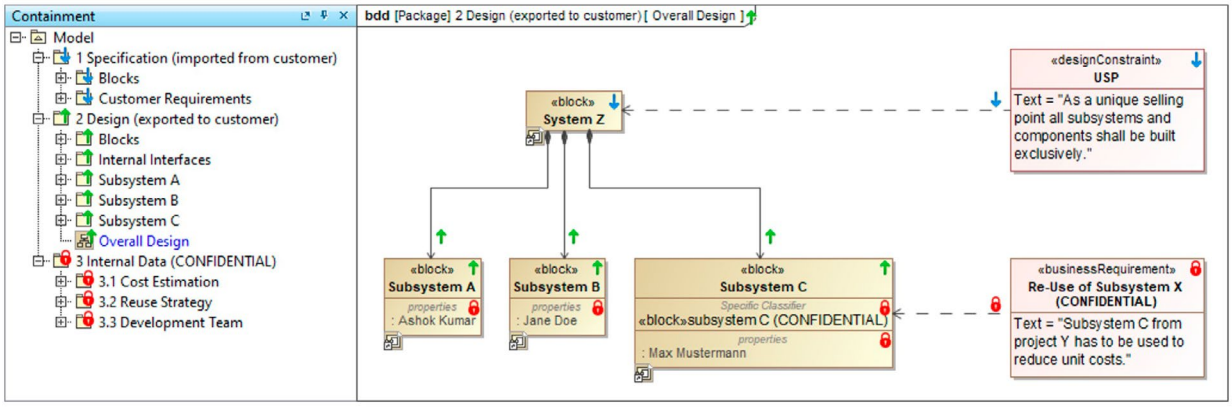


Figure 15: Example IP protection based on SysML packages

IP protection becomes complicated when simply leaving away internal elements during export would lead to incomplete diagrams or relations with dead ends. In these conflicting cases, internal elements have to be obfuscated in an appropriate way for export. This conflict and its resolution has to be reported by the exporting tool to the exporting user in order to avoid confusion.

In the following chapters, the use cases „Requirements Engineering & System Design" and "Verification & Validation" are described. They extend the use case "Model Exchange" with the need to exchange additional information between companies in combination with system models.

3.2 Use Case "Requirements Engineering & System Design"

The Use Case "Requirements Engineering & System Design" is discussed inside the Work Package 6 (WP6) of the SysML WF. Like in WP4, also in WP6 the use case describes the exchange of system model information between two companies (e.g. an OEM and Supplier), but in this case extended to the need of exchanging linked requirement information.

The scenario overview in the following Figure 16 shows the main activities that should be considered for the implementation inside the SysML IF. It is based on the process of the ReqIF WF. The scenario overview contains different sub-scenarios describing the collaboration between OEM and supplier. Depending on the collaboration approach,

it might be that a supplier only reviews the requirements and system model and adds comments, but it might also be that the supplier needs to update or add content inside the requirement specification or the system model. Both options should be considered.

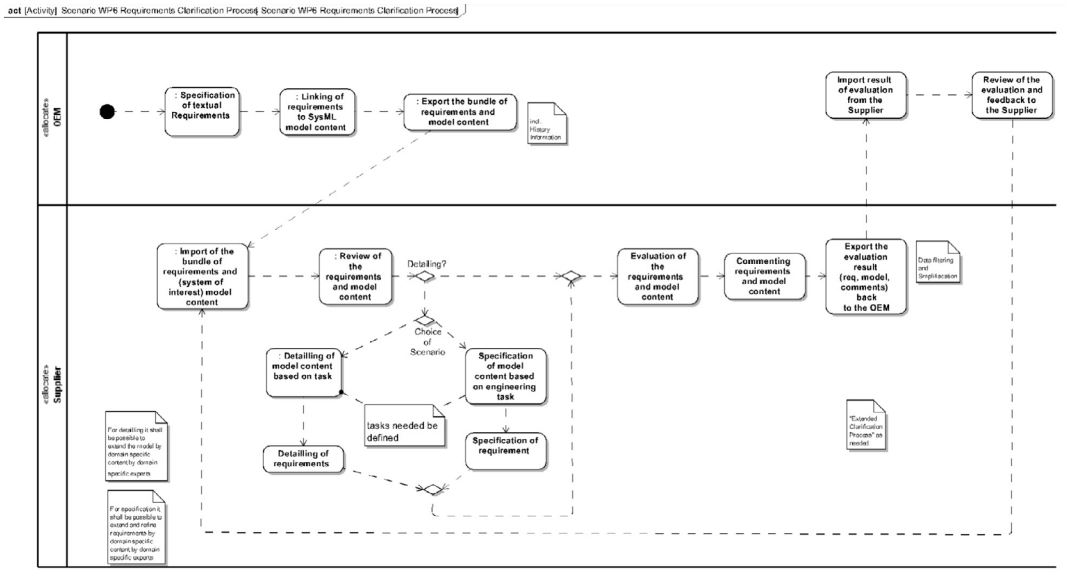


Figure 16: Collaboration scenario: WP6 "Requirements Engineering"

Requirements engineering is a key activity of MBSE and the traceability between requirements and system models is essential. But in most companies that follow an MBSE approach the IT tool landscape contains separate tools for requirements management and system modeling, due to the specific capabilities of each tool.

How to link requirements and system models from a technical perspective is a question, that is relevant for many companies and can be answered in different ways, e.g.:

- **Importing requirements into the system model via ReqIF:** The easiest solution might be, to import requirements into the system model. Most of current available system modelers offer an import mechanism that can read at least ReqIF files and create requirement objects inside the system model. This approach supports of course only a one-directed import and demands an increased amount of manual work to keep the requirements synchronized between requirements management tool and system modeler.
- **Use of synchronization plugins / addons:** Some system modelers offer additional plugins and addons, that allow the synchronization between system modeler and requirements management tool via native or standardized interfaces. This approach allows often a bidirectional synchronization, offers a better user experience and reduces the manual effort for the synchronization of requirements between the different tools. In fact, this approach depends on the availability of plugins for the needed tool combinations
- **Use of a traceability platform:** Another approach is the usage of an external traceability platform that manages the links between requirements and system model artefacts. The advantage of this approach is, that traceability platforms often offer a higher number of supported tool combinations and therefore can be used for extending the traceability also across other tools and disciplines. On the other hand, these tools often demand to leave the work environment to create links and also to analyze the traceability.

Depending on the selected approach, the exchange of system models in combination with linked requirements demands different technical implementations of an exchange solution. The high-level requirements that have been addressed by the SysML WF towards the SysML IF are the following:

Name	Text
Atomic traces between requirements and model elements	The requirements (documented outside the SysML tool) shall be traceable to the relevant model content. (relevant model content: e.g. diagrams, elements and traces between elements). The linking of the requirements to the model elements shall be tool-versions stable.
Versioning of traces between requirements and model elements	For the configuration management the atomic elements (model elements) shall be versionable. The traces shall be version dependent. The traces between the new (changed) version of the element shall be markable as suspected.
Model-based requirements exchange	We need to exchange model content as requirements (model content: diagrams (use case, state machine, activity diagrams, sequence diagrams and package diagrams, elements and traces between elements) between collaborating automotive companies based on a requirements clarification process.
Model-supported requirements exchange	We need to exchange system model content linked to textual requirements between collaborating automotive companies based on the established requirements clarification process.

Table 4: Overview High-Level Requirements WP6 "Requirements Engineering"

3.3 Use Case "Verification & Validation"

The Use Case "Verification & Validation" is discussed inside the Work Package 7 (WP7) of the SysML WF. Like the two previous use cases, this use case refers to an exchange scenario, focusing on the aspect of V&V artefacts linked to the system model.

The scenario is an extension of the scenario considered in WP6 (Requirements Engineering). It is assumed that there exists already traceability between requirements and the system model and based on this knowledge, verification cases can be defined and linked towards the requirements. With this approach the system model becomes the central data hub for linking information from the top left side of the V-Model (Requirements Engineering) towards the top right side of the V-Model (Verification & Validation). Information that is contained inside the system model (e.g. description of interfaces, system behavior, …) can be linked and used inside requirements and test cases.

The discussions inside WP7 showed that most companies are still some steps away from following such an approach, but it will become more and more important in the future. That's why the initial scenario that was considered, was limited to the scope of creating and exchange coverage reports between an OEM and a supplier. The scenario describes that a supplier performs verification activities based on the system model and the requirements received from the OEM, and then adds and links coverage information towards the system model and requirements, that can later be imported by the OEM again, to gain information on the coverage status of his requirements.

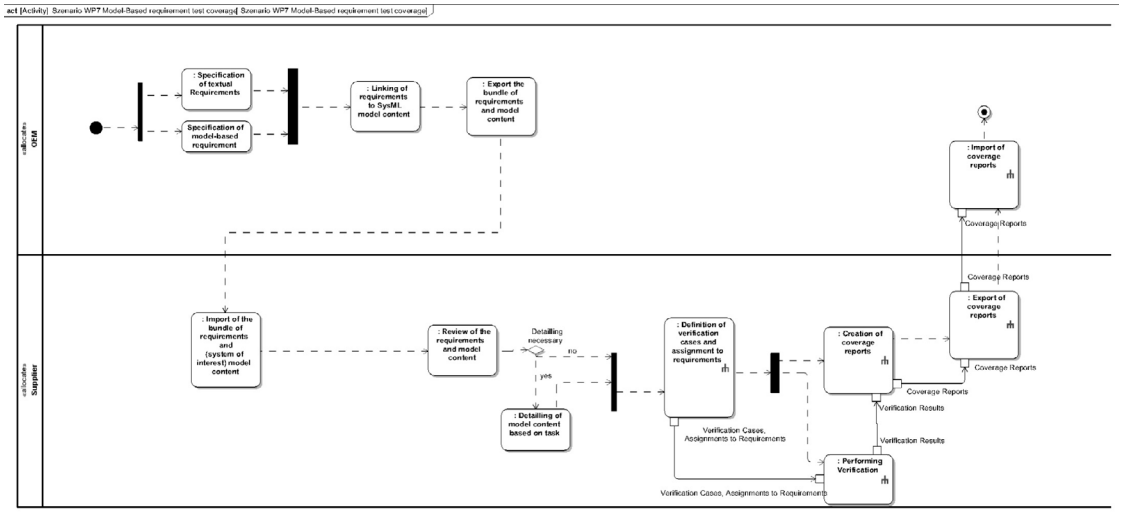


Figure 17: Collaboration scenario WP7 “Verification & Validation”

4 SysML Demo Model

The SysML demo model that was taken as an example for the SysML WF/IF is the exemplary case of a Hybrid Sport Utility Vehicle (HSUV). Such a system presents interesting aspects for modelling, as it can be complex with several domains and contains requirements with opposing nature, e.g., fuel/power efficiency vs. off-road capability. On account of this, the model focuses on requirements, performance analyses, structure and behavior of the HSUV system, concentrating in specific on the power subsystem. This HSUV example which was published originally by the OMG (https://www.omg.org/spec/SysML/1.2/PDF) was extended by the SysML WF and therefore the example model is called eHSUV (extended Hybrid SUV). The goal of the extension was to have all SysML elements inside the model, that are relevant for the model exchange.

Figure 18 shows the overall model structure represented in a package diagram where the main focus aspects of the model can be seen:

- Context (Use Cases and Boundaries)
- Behavior
- Structure
- Requirements and Analysis

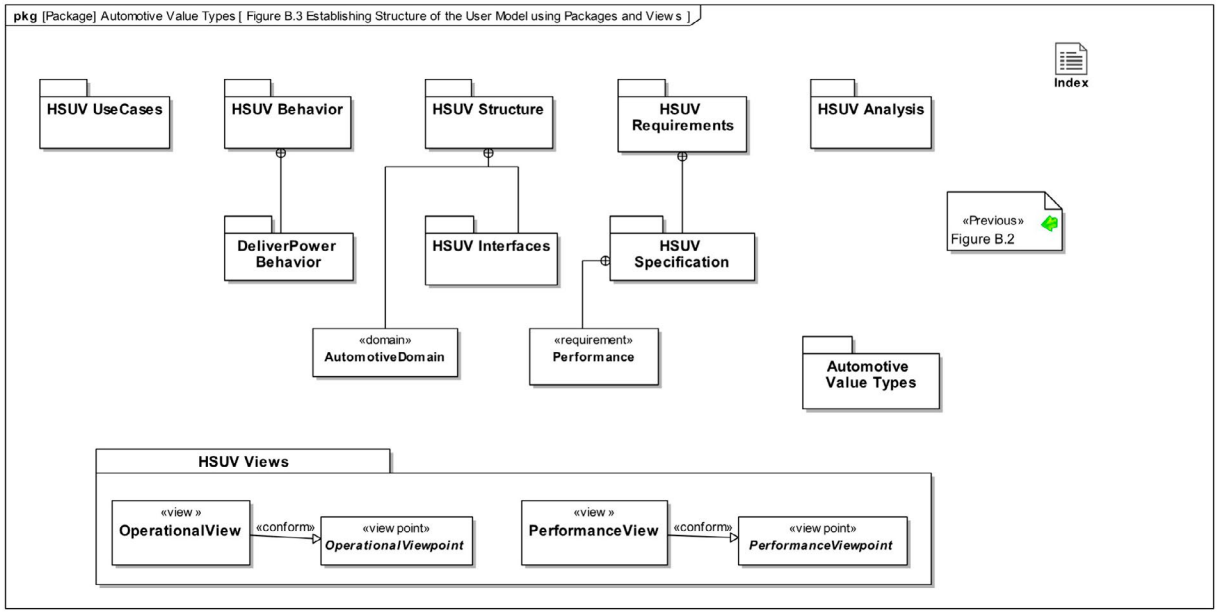


Figure 18: Package diagram showing the model structure and its views

The contents and organization of these packages is described next.

4.1 Use Cases and Boundaries

For this aspect only top level use cases like “operate”, “maintain”, “register” or “insure” the vehicle are depicted and allocated towards the actors (driver, maintainer, insurance company, etc.), as seen in Figure 19. Use cases described from an operational perspective like drive and park the vehicle, exist in the model as well as its sub use cases like accelerate and brake in order to move in the direction of the objective: describing the power subsystem.

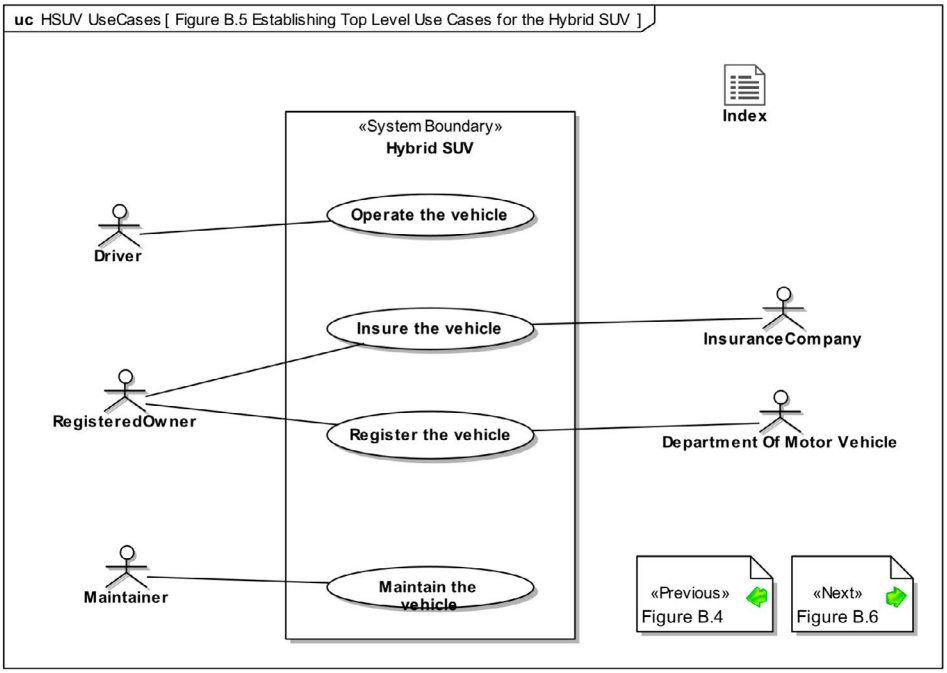


Figure 19: Use Case diagram example of eHSUV

Boundaries are also set in user-defined context diagrams using an internal block diagram as base. The context elements as weather, road and vehicle cargo are defined as external elements thus limiting the system boundaries, see Figure 20.

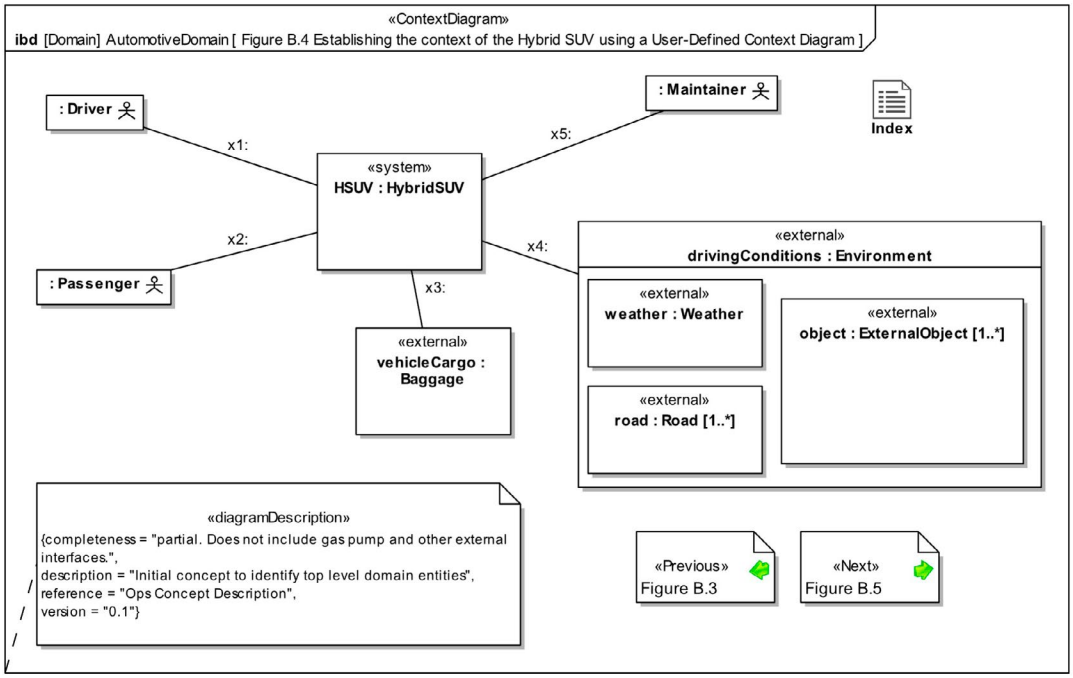


Figure 20: Context diagram of eHSUV

4.2 Behavior

In case of the behavior, the model uses interaction elements in sequence diagrams, such as brake, steer, accelerate, etc. to describe the drive action as a black box and also the start vehicle action as a white box. Furthermore, state elements such as off, braking, idle, etc., are modelled and presented in state machine diagrams for the describing the operational states. Activities are used to describe the behavior of system and are modeled as part of different activity diagrams, representing functions of system, see Figure 21.

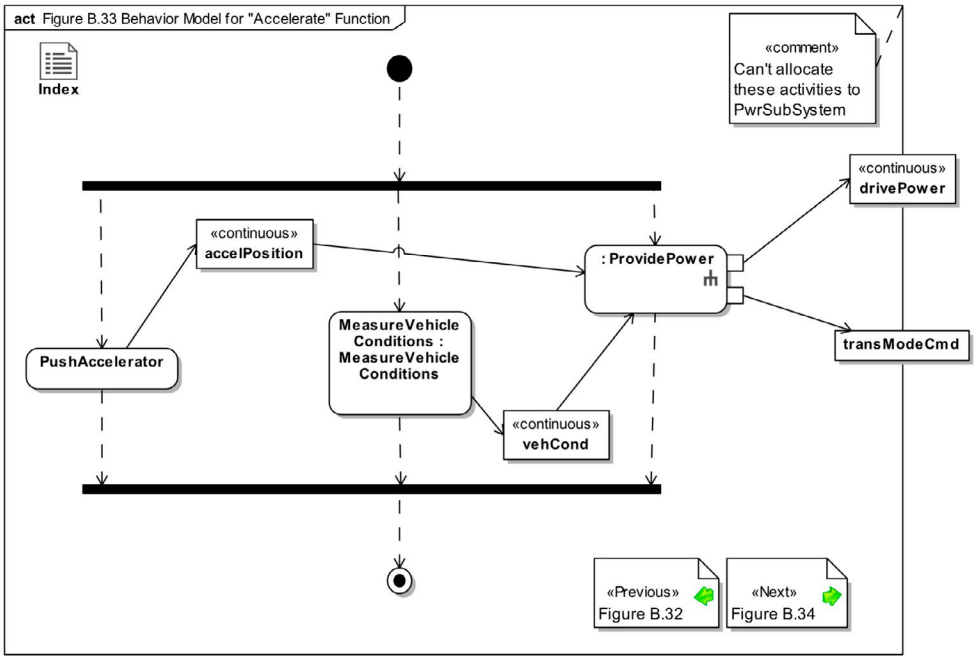


Figure 21: Activity Diagram for the Accelerate Function

4.3 Requirements

For the requirements and their respective sub requirements, typical tables and diagrams with derived and satisfied relations to other requirements and blocks are used. An excerpt of requirements is shown in the following Figure 22.

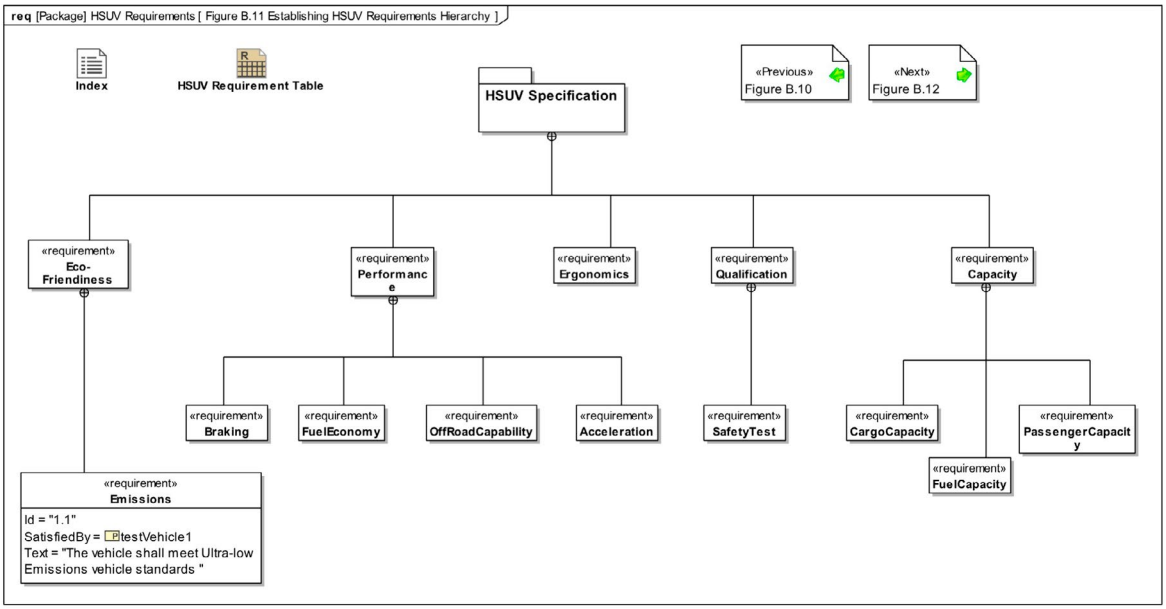


Figure 22: A requirement diagram depicting their hierarchy with an example of a textual requirement

4.4 Structure

The structural description of the model takes place first using block definition diagrams for describing the upper level domain with elements already mentioned like actor and external components, see Figure 23.

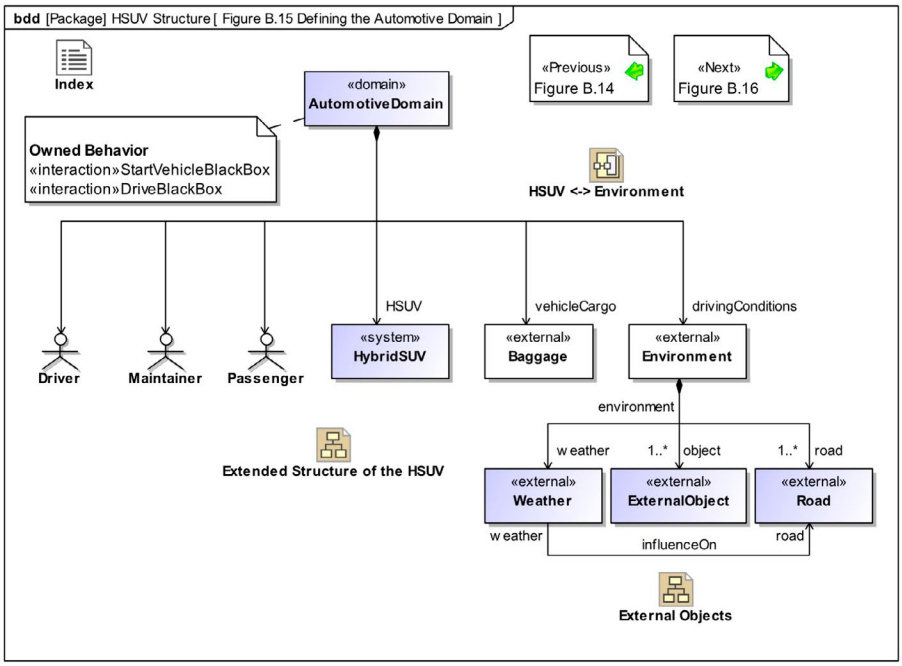


Figure 23: The upper level domain described in a BDD

Going deeper into the system, the subsystems of the eHSUV are modelled and presented in a diagram shown in Figure 24.

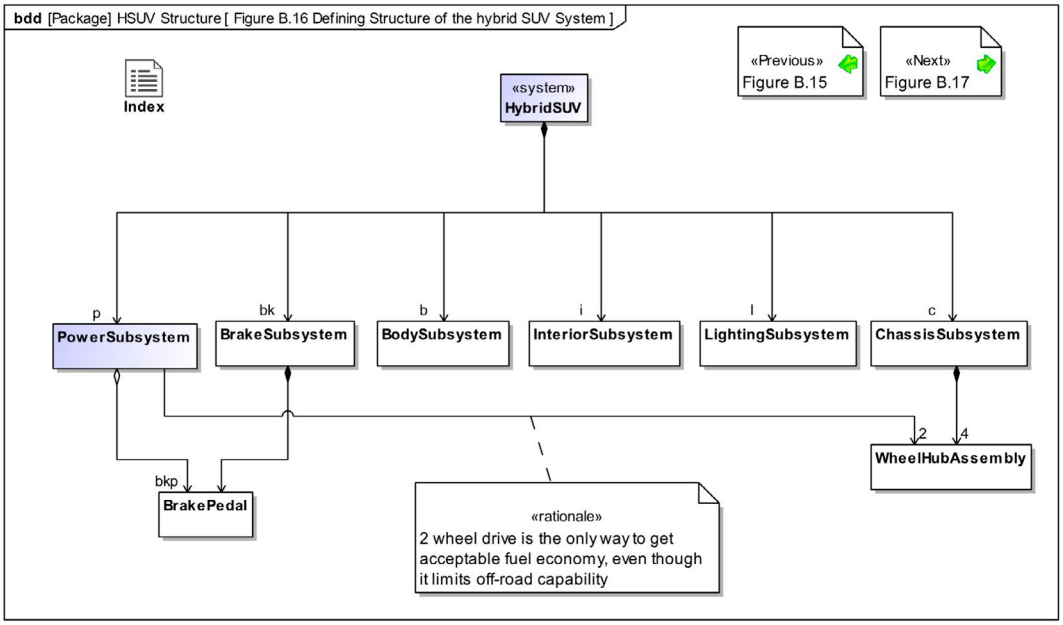


Figure 24: Breakdown of the subsystem of the eHSUV model.

Internal block diagrams describing the connections between the subsystems are also modeled. One example is the Power Subsystem as part of the main focus of the model, and can be seen in Figure 25.

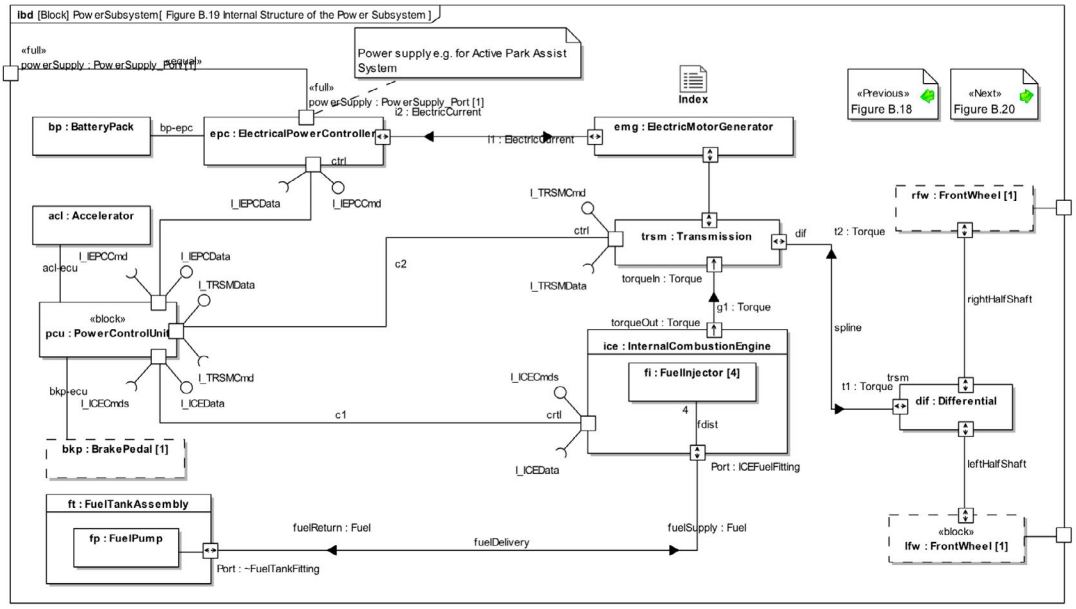


Figure 25: Internal Block Diagram of Power Subsystem

4.5 Analysis

In order to analyze the power subsystem of the model, and more precisely its performance, several elements and diagrams are used. Elements like viewpoints, constraint blocks and even a stereotype "measure of effectiveness" that is user defined, are used in block definition, package and also parametric diagrams in order to assess key aspects of the performance such as fuel economy and vehicle dynamics. An impression of how detailed and complex this parametric analysis can be, the fuel economy diagram is seen in Figure 26.

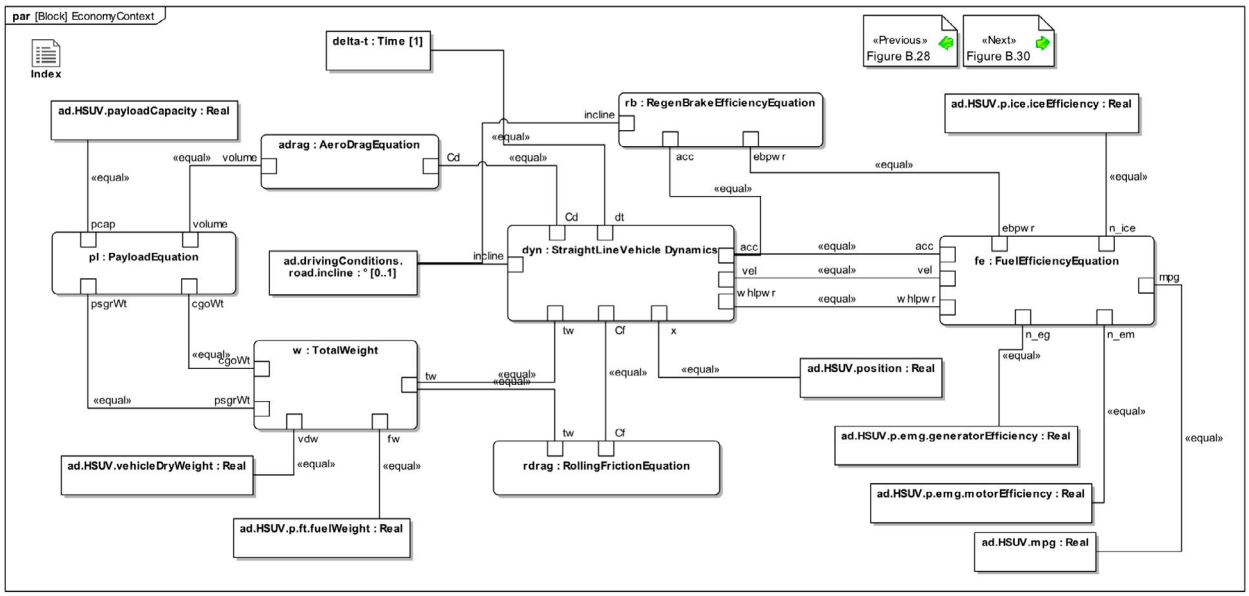


Figure 26: Parameter Diagram of eHSUV example

4.6 Summary

The eHSUV Model presents a comprehensive example with a defined focused objective. It provides all the necessary base information and elements to establish the model and works its way through to the deeper levels in order to analyze its objective: assessing the power subsystem and its efficiency. It achieves this by defining and using only what is fundamental to the eHSUV and also showing how to convey information in selected diagrams. Furthermore, it succeeds in demonstrating the utilization of user defined elements. All these points make the eHSUV model to an appropriate test model for the different demonstrators, analyzing various aspects of a SysML based system model.

5 SysML IF Demonstrators

Based on the described scenarios and the requirements defined by the SysML WF, the vendors participating inside the SysML IF presented different solutions and approaches for the addressed use cases. The following chapters give an overview on the formats that were considered so far, as well as on the different demonstrators.

5.1 Overview of Existing Exchange Formats

As an introduction an overview of the existent formats and approaches is given, that are from the current point of view closest to an exchange standard for system models based on SysML. In specific these are:

- XMI
- SpecIF
- Project MTIP

So far, no official benchmark has been performed to compare the different formats and approaches, but some of the findings that were already made are mentioned in the following chapters.

5.1.1 XMI

There are a number of ways to exchange information and also meta data in the context of system modeling and especially systems engineering for example via XML-Metadata-Interchange (XMI) and also the ISO 10303 AP233. These are only two possibilities to exchange data. The following section will focus on the XMI approach. The XMI stands for XML-Metadata-Interchange, so it is a format for the exchange of Extensible Markup Language (XML) data and is standardized by the Object Management Group (OMG). The XML-Metadata-Interchange is a very common format for the exchange of XML data, data in the context of UML or also data which can be expressed with the help of the Meta Object Facility (MOF), which is also a standard of the OMG. The latest XMI Version is the 2.5.1 and was published in June 2015.

Common SysML modeling tools support the XMI-Format, for a tool neutral exchange of SysML models, but with a lack of information. The possibilities with the exchange of SysML models along the XMI standard, will be described in the following section.

First some aspects about XMI, which are defined by the standard:

- A representation of objects using XML elements and attributes.
- Is the standard mechanism for linking objects in the same file or between files.
- A validation of the XMI documents using the XML schema specified by the OMG.
- An object identity that gives you the freedom to create a reference from other objects.
- Each XMI file must meet certain requirements to comply with the format:
- All elements and attributes must be imported and must not be copied directly into the schema.
- All model constructs have an element declaration.
- Any extension to the XML model must be mentioned and documented in the schema.

All the above requirements must be met in order to obtain a satisfactory result from the exchange. Also, the exchange itself brings three requirements with it from the specification. These were also defined like all other requirements by the OMG and fixed as standard.

- All important aspects of metadata should part of the XML document and can also be recovered from it.
- An XML document should be as compact as possible but at the same time as complete as possible and in no case should there be any loss of information.
- The XML document should reflect the model completely.

When creating an XMI document, there are some tools that help to describe the document and the model. They are very simple, but they are sufficient to describe the XMI completely. These four tools are:

- XMI.difference
- XMI.delete
- XMI.add
- XMI.replace

The main tool here is the XMI.difference it is responsible for showing the difference between the original model and the generated XML document. With XMI.delete a delete from the original model is defined. Replacing an element in the initial model is done with XMI.replace, but no element is deleted, only replaced and is therefore no longer part of the container. As a last command, XMI.add is available to the user. With this command other elements, which did not exist in the original model, can be added. (OMG, https://www.omg.org/spec/XMI/About-XMI/, 2015)

View on the XMI exchange possibilities

To check the status of the possibilities of the XMI exchange, two partners (VPE & Koneksys) were assigned. For the check-up four different SysML modeling tools were used to test the exchange of SysML models via XMI between modeling applications: IBM Rhapsody, Cameo (MagicDraw), PTC Integrity Modeler, and Sparx Systems Architect. Complete interoperability without data loss was not achieved when converting SysML models between tools. Most data exchange operations maintained the UML-related information contained in SysML models but lost the SysML-specific information. A complete overview of the results is presented in Figure 22, thereby all usually used model elements are represented. The result can be interpreted as follows:

- ✓ Transfer correctly
- × not transferred / incorrectly transferred
- O with errors / incompletely transferred

According to the UML standard: "It must be possible to interchange profiles between tools, together with models to which they have been applied". Unfortunately, in practice, this is not the case. SysML models are expected to be interchanged using the XMI standard. XMI is an OMG standard and can be used for any data whose metamodel can be expressed in Meta-Object Facility (MOF), which is another OMG standard. The current version of SysML is currently implemented as a UML profile. UML has a metamodel expressed in MOF. UML models can therefore be described in XMI. However, SysML does not have a metamodel expressed in MOF. The SysML standard is defined as an extension of UML. A SysML model therefore consists of UML elements which are extended by SysML stereotypes. In other words, a SysML model is created by applying SysML stereotypes to UML elements.

Three options exist to explain the lack of interoperability of SysML based on XMI:

1. Lack of support for XMI by SysML vendors
2. Unclear serialization of profiles and stereotype applications in XMI
3. Combination of both points above

Option #1 seems to be supported by the frequent use of the *xmi:extension* tag by vendors to describe SysML model information in XMI. The *xmi:extension* tag is designed to contain extended information outside the scope of the SysML model. The *xmi:extension* tag is not meant to support data interchange as it can contain arbitrary XML content. It is unclear at this point why vendors choose to frequently use the *xmi:extension* tag to describe SysML information in XMI.

Option #2 was raised as an OMG issue which is currently unresolved for UML 2.6 Revision Task Force. It cites the UML specification including following quote: "A profile is an instance of a UML2 metamodel, not a CMOF metamodel. Therefore, the MOF to XMI mapping rules do not directly apply for instances of a profile. Figure 27 is an example of a mapping between a UML2 Profile and an equivalent CMOF model. This mapping is used as a means to explain and formalize how profiles are serialized and exchanged as XMI. Using this Profile to CMOF mapping, rules for mapping CMOF to XMI can be used indirectly to specify mappings from Profiles to XMI.". A discussion related to this issue can be found at OMG.

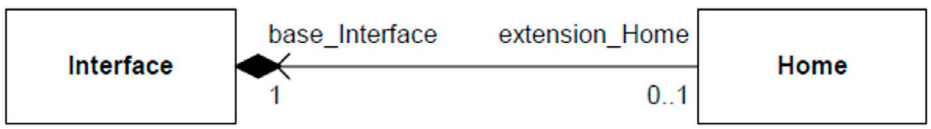
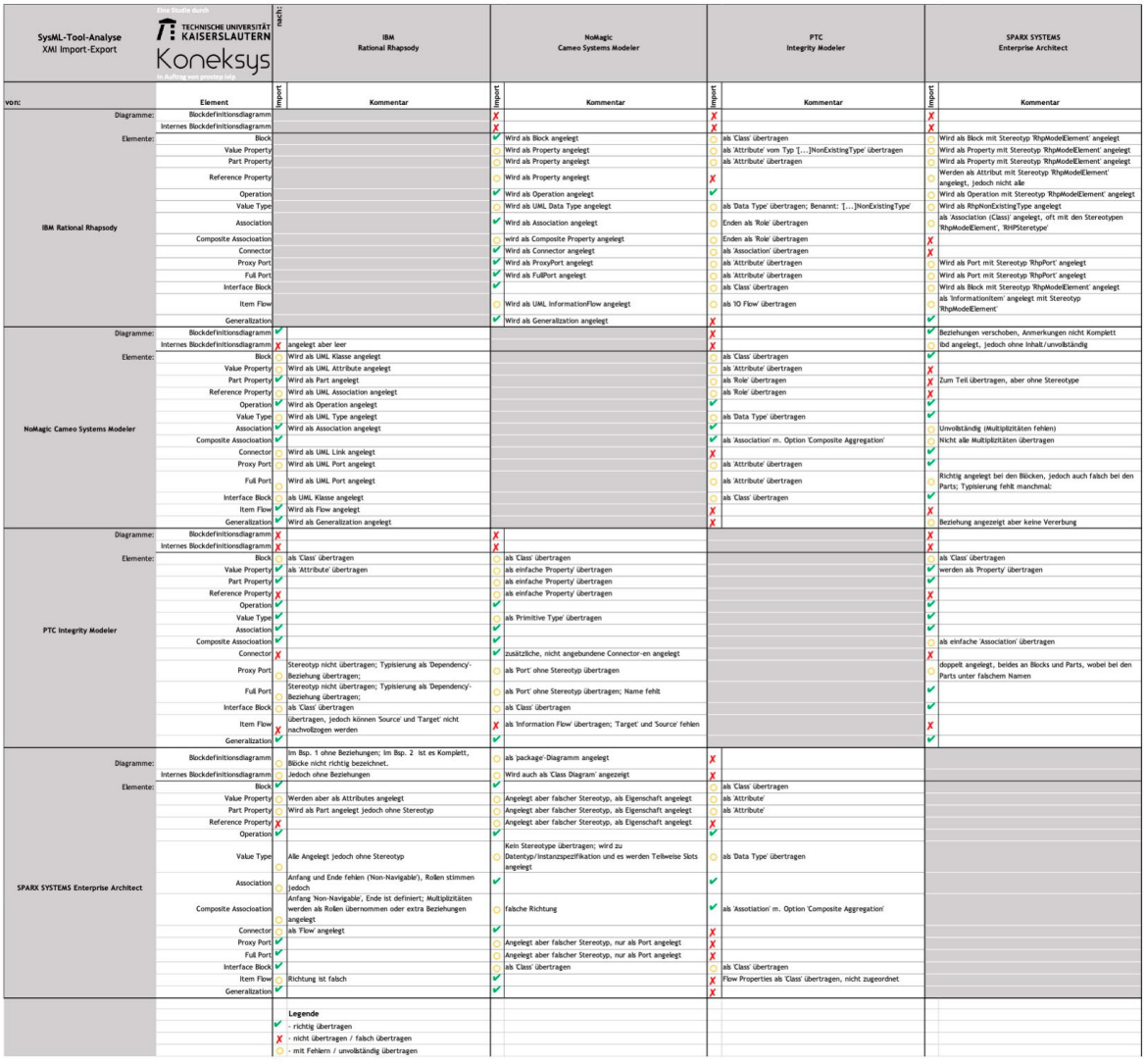


Figure 27: MOF model equivalent [UML]

Based on these two observations, it seems that the lack of interoperability is most likely resulting from a combination of option #1 and #2.

Currently in 2022 the problem is also mentioned by the OMG, which has the effect that the vendors planed some future work on the implementation of the XMI standard.



SysML-Tool-Analyse XMI Import-Export		nach:	IBM Rational Rhapsody		NoMagic Cameo Systems Modeler		PTC Integrity Modeler		SPARX SYSTEMS Enterprise Architect	
von:	Element	Import	Kommentar	Import	Kommentar	Import	Kommentar	Import	Kommentar	
Diagramme:	Blockdefinitionsdiagramm			✗		✗		✗		
	Internes Blockdefinitionsdiagramm			✗		✗		✗		
Elemente:	Block			✓	Wird als Block angelegt	○	als 'Class' übertragen	○	Wird als Block mit Stereotyp 'RhpModelElement' angelegt	
	Value Property			○	Wird als Property angelegt	○	als 'Attribute' vom Typ '[…]NonExistingType' übertragen	○	Wird als Property mit Stereotyp 'RhpModelElement' angelegt	
	Part Property			○	Wird als Property angelegt	○	als 'Attribute' übertragen	○	Wird als Property mit Stereotyp 'RhpModelElement' angelegt	
	Reference Property			○	Wird als Property angelegt	✗		○	Werden als Attribut mit Stereotyp 'RhpModelElement' angelegt, jedoch nicht alle	
	Operation			✓	Wird als Operation angelegt	✓		○	Wird als Operation mit Stereotyp 'RhpModelElement' angelegt	
	Value Type			○	Wird als UML Data Type angelegt	○	als 'Data Type' übertragen; Benannt: '[…]NonExistingType'	○	Wird als RhpNonExistingType angelegt	
IBM Rational Rhapsody	Association			✓	Wird als Association angelegt	○	Enden als 'Role' übertragen	○	als 'Association (Class)' angelegt, oft mit den Stereotypen 'RhpModelElement', 'RHPStereotype'	
	Composite Association			○	wird als Composite Property angelegt	○	Enden als 'Role' übertragen	✗		
	Connector			✓	Wird als Connector angelegt	○	als 'Association' übertragen	✗		
	Proxy Port			✓	Wird als ProxyPort angelegt	○	als 'Attribute' übertragen	○	Wird als Port mit Stereotyp 'RhpPort' angelegt	
	Full Port			✓	Wird als FullPort angelegt	○	als 'Attribute' übertragen	○	Wird als Port mit Stereotyp 'RhpPort' angelegt	
	Interface Block			✓		○	als 'Class' übertragen	○	Wird als Block mit Stereotyp 'RhpModelElement' angelegt	
	Item Flow			○	Wird als UML InformationFlow angelegt	○	als 'IO Flow' übertragen	○	als 'Informationitem' angelegt mit Stereotyp 'RhpModelElement'	
	Generalization			✓	Wird als Generalization angelegt	✗		✓		
Diagramme:	Blockdefinitionsdiagramm	✓				✗		✓	Beziehungen verschoben, Anmerkungen nicht Komplett	
	Internes Blockdefinitionsdiagramm	✗	angelegt aber leer			✗		○	ibd angelegt, jedoch ohne Inhalt/unvollständig	
Elemente:	Block	○	Wird als UML Klasse angelegt			○	als 'Class' übertragen	✓		
	Value Property	○	Wird als UML Attribute angelegt			○	als 'Attribute' übertragen	✗		
	Part Property	✓	Wird als Part angelegt			○	als 'Role' übertragen	✗	Zum Teil übertragen, aber ohne Stereotype	
	Reference Property	○	Wird als UML Association angelegt			○	als 'Role' übertragen	✗		
	Operation	✓	Wird als Operation angelegt			✓		✓		
	Value Type	○	Wird als UML Type angelegt			○	als 'Data Type' übertragen	✓		
NoMagic Cameo Systems Modeler	Association	✓	Wird als Association angelegt			✓		○	Unvollständig (Multiplizitäten fehlen)	
	Composite Association	✓				✓	als 'Association' m. Option 'Composite Aggregation'	○	Nicht alle Multiplizitäten übertragen	
	Connector	○	Wird als UML Link angelegt			✗		✓		
	Proxy Port	○	Wird als UML Port angelegt			○	als 'Attribute' übertragen	✓		
	Full Port	○	Wird als UML Port angelegt			○	als 'Attribute' übertragen	○	Richtig angelegt bei den Blöcken, jedoch auch falsch bei den Parts; Typisierung fehlt manchmal:	
	Interface Block	○	als UML Klasse angelegt			○	als 'Class' übertragen	✓		
	Item Flow	✓	Wird als Flow angelegt			✗		✗		
	Generalization	✓	Wird als Generalization angelegt			✗		○	Beziehung angezeigt aber keine Vererbung	
Diagramme:	Blockdefinitionsdiagramm	✗		✗				✗		
	Internes Blockdefinitionsdiagramm	✗		✗				✗		
Elemente:	Block	○	als 'Class' übertragen	○	als 'Class' übertragen			○	als 'Class' übertragen	
	Value Property	✓	als 'Attribute' übertragen	○	als einfache 'Property' übertragen			✓	werden als 'Property' übertragen	
	Part Property	✓		○	als einfache 'Property' übertragen			✓		
	Reference Property	✗		○	als einfache 'Property' übertragen			✗		
	Operation	✓		✓				✓		
	Value Type	✓		○	als 'Primitive Type' übertragen			✓		
PTC Integrity Modeler	Association	✓		✓				✓		
	Composite Association	✓		✓				○	als einfache 'Association' übertragen	
	Connector	✗		✓	zusätzliche, nicht angebundene Connector-en angelegt			✗		
	Proxy Port	○	Stereotyp nicht übertragen; Typisierung als 'Dependency'-Beziehung übertragen;	○	als 'Port' ohne Stereotyp übertragen			○	doppelt angelegt, beides an Blocks und Parts, wobei bei den Parts unter falschem Namen	
	Full Port	○	Stereotyp nicht übertragen; Typisierung als 'Dependency'-Beziehung übertragen;	○	als 'Port' ohne Stereotyp übertragen; Name fehlt			✓		
	Interface Block	○	als 'Class' übertragen	○	als 'Class' übertragen			✓		
	Item Flow	✗	übertragen, jedoch können 'Source' und 'Target' nicht nachvollzogen werden	✗	als 'Information Flow' übertragen; 'Target' und 'Source' fehlen			✗		
	Generalization	✓		✓				✓		
Diagramme:	Blockdefinitionsdiagramm	○	Im Bsp. 1 ohne Beziehungen; im Bsp. 2 ist es Komplett, Blöcke nicht richtig bezeichnet.	○	als 'package'-Diagramm angelegt	✗				
	Internes Blockdefinitionsdiagramm	○	Jedoch ohne Beziehungen	○	Wird auch als 'Class Diagram' angezeigt	✗				
Elemente:	Block	✓		✓		○	als 'Class' übertragen			
	Value Property	○	Werden aber als Attributes angelegt	○	Angelegt aber falscher Stereotyp, als Eigenschaft angelegt	○	als 'Attribute'			
	Part Property	○	Wird als Part angelegt jedoch ohne Stereotyp	○	Angelegt aber falscher Stereotyp, als Eigenschaft angelegt	○	als 'Attribute'			
	Reference Property	✗		○	Angelegt aber falscher Stereotyp, als Eigenschaft angelegt	✗				
	Operation	✓		✓		✓				
	Value Type	○	Alle Angelegt jedoch ohne Stereotyp	○	Kein Stereotype übertragen; wird zu Datentyp/Instanzspezifikation und es werden Teilweise Slots angelegt	○	als 'Data Type' übertragen			
SPARX SYSTEMS Enterprise Architect	Association	○	Anfang und Ende fehlen ('Non-Navigable'), Rollen stimmen jedoch	✓		✓				
	Composite Association	○	Anfang 'Non-Navigable', Ende ist definiert; Multiplizitäten werden als Rollen übernommen oder extra Beziehungen angelegt	○	falsche Richtung	✓	als 'Assotiation' m. Option 'Composite Aggregation'			
	Connector	○	als 'Flow' angelegt	✓		✗				
	Proxy Port	✓		○	Angelegt aber falscher Stereotyp, nur als Port angelegt	✗				
	Full Port	✓		○	Angelegt aber falscher Stereotyp, nur als Port angelegt	✗				
	Interface Block	✓		○	als 'Class' übertragen	○	als 'Class' übertragen			
	Item Flow	○	Richtung ist falsch	✓		✗	Flow Properties als 'Class' übertragen, nicht zugeordnet			
	Generalization	✓		✓		✗				

Legende
✓ - richtig übertragen
✗ - nicht übertragen / falsch übertragen
○ - mit Fehlern / unvollständig übertragen

Figure 28: Results of the SysML-Model exchange via XMI-Standard

5.1.2 SpecIF

Introduction

The Specification Integration Facility (SpecIF) (Specification Integration Facility (SpecIF)) is being developed as an open standard by the GfSE as an aid for collaboration in model-based systems engineering (MBSE) (Kaufmann, 2015) (Dungern O. v., Kollaboration im Systems Engineering mit ReqIF, April 2022). It is no more and no less than an **information model** constituting a semantic net. It is applicable to any notation, technology or tool. It does not replace a notation or a specific meta-model, nor can it be replaced by any of those (Dungern O. v., November 2020).

The SpecIF information model is built on OMG EMOF (Instances ← Classes ← SpecIF ← EMOF) (Dungern O. v., Semantic Model Integration for System Specification, 2016). So far, SpecIF has been implemented using JSON-Schema, while it can equally be implemented in the XML world.

SpecIF could be used for all three use cases outlined in Chapter 3: "Model-Exchange", "Requirements Engineering & System Design" and "Verification & Validation".

The advantage of adopting SpecIF for a given use-case is that the same transformations, persistence layers, viewers and editors can be employed, no matter which domain specific type-set or vocabulary is used. All SpecIF data-sets ("projects") can be combined to a single semantic net ("graph") which can be consistently queried with existing query-languages. As graph-data, quasi unlimited scalability and superior performance is ensured.

Purpose

Experience has led the makers of SpecIF to assume:

- There will be always specialized tools for different purposes.
- It is unwise to require collaborators to use certain tools or even a single tool.
- Yet, there is an interest
  - to navigate, search and audit partial results in a common context to exchange model information between organizations and tools.

That is where SpecIF kicks in: Ensure that different organizations using different methods and tools can effectively collaborate when engineering a common product.

Model Integration

Merging requirements with a system model is an important capability needed for the use-case "Requirements Engineering & System Design". For this use case SpecIF offers a concept for integrating different formats into SpecIF and add semantic information. The SpecIF container can then be serialized again into ReqIF and imported into existing tools. But currently most tools are not capable of reading the additional semantic information out of the ReqIF, that could be added via SpecIF.

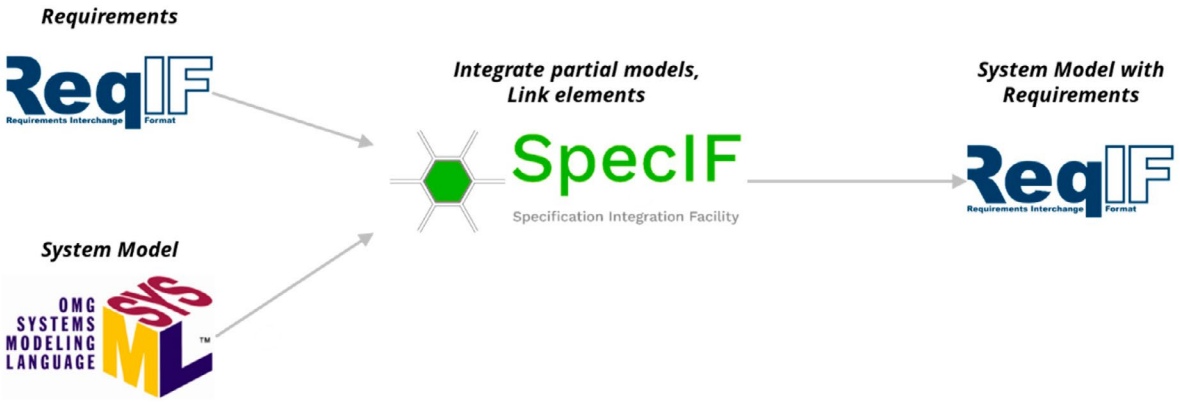


Figure 29: SpecIF approach for Use Case "Requirements Engineering & System Design"

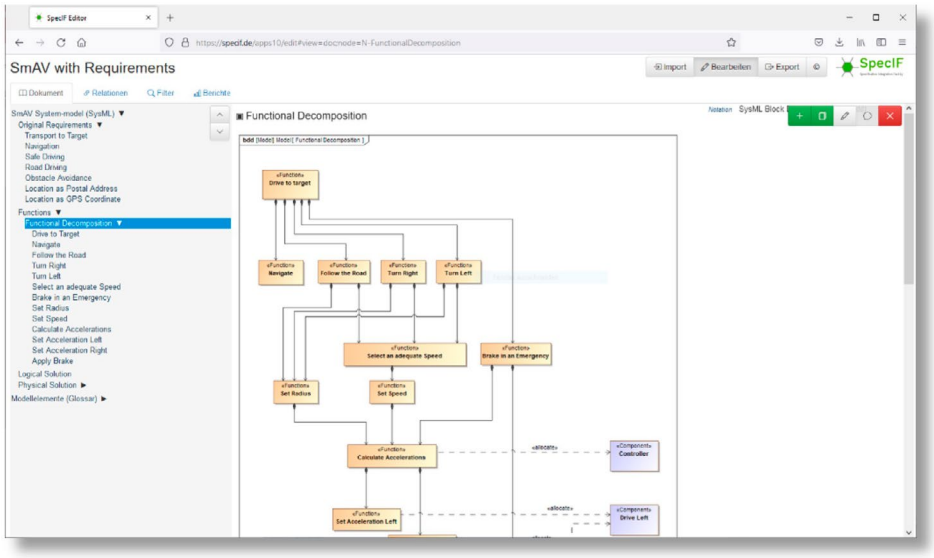
Most importantly, model integration depends on a suitable abstraction and the use of a common vocabulary (Dungern O. v., Integration von Systemmodellen mit fünf fundamentalen Elementtypen., November 2015).

Example

The following screenshots are made with a reference implementation of a *SpecIF Model Integrator and Editor*. At the time of writing, it imports and integrates SpecIF, ReqIF, Archimate, BPMN-XML and Excel files and exports models in SpecIF, HTML with embedded SpecIF, ReqIF, Turtle (experimental), ePub v2 and WORD-ML format. While it is not an enterprise-grade software implementation, it is being used successfully in industry projects.

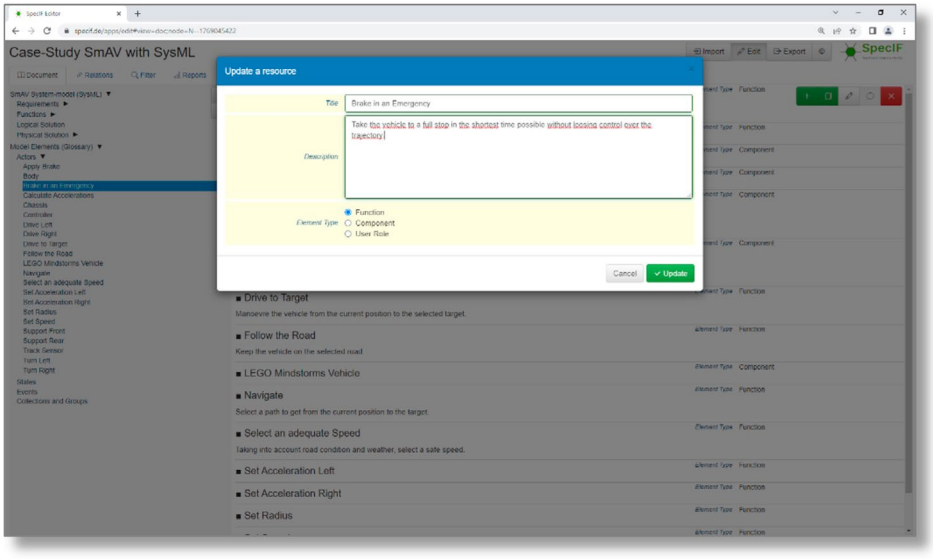
The model shown is a "Small Autonomous Vehicle", prepared by the Fachhochschule Esslingen, Prof. Dr. Ralf Schuler and his students. Here, it is a manually transformed SysML-Model authored using Cameo. While automatic SpecIF converters have been made available for several notations and tools, the transformation from and to SysML is yet to complete.

Model Diagrams:



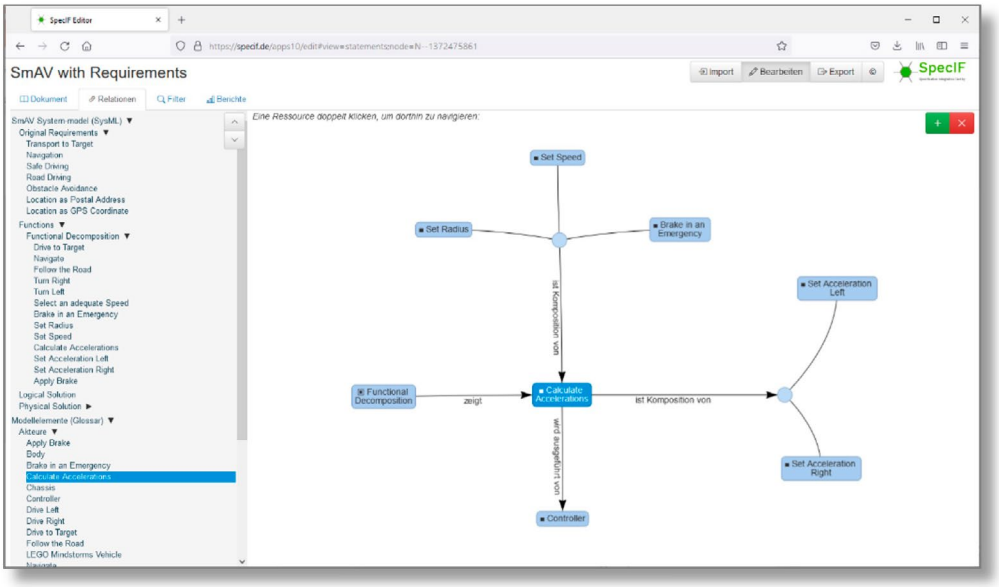
The original model diagrams are transformed to resources poin ting to SVG, PNG or JPG images. Any shown model element is equally a resource, where a SpecIF:shows relation is established. So, a diagram 'knows' which elements are shown and a given element 'knows' on which diagrams it app ears. All shown relations are also represented in SpecIF, here the classes SysML:isComposedOf and SysML:isAllocatedTo apply. If diagrams are supplied as SVG, shown graphic elements can be hovered or clicked to show addi tional information or navigate within the model.

Model-Elements ("Resources", Nodes):



Model-Elements are the resources or nodes. Several resource classes with an individual set of properties can be defined. SpecIF classes can be used to build the user-interface: For example, a dialog for editing resource properties may show a field per defined property.
Here, the resource to edit has three properties, the last one having enumerated values. Input validation can be automatically applied according to a properties' data type.
Thus, the tool is very simple, if data with a simple set of classes is at hand, while it appears more elaborated in other cases.

Relations between Model-Elements ("Statements", Edges):



Next to the resource properties, the relations between model-elements, the statements or edges, convey meaning. Statements can be visible on dia-grams or hidden. The screenshot shows all relationships of the selected model-element, no matter on which diagram it is visible or not at all; it is a piece of the semantic net.
The statement classes may define permissible classes for resources to be used as subjects or objects, thus assuring that only meaningful statements can be defined.
The relations allow for model checking using logical or heuristic rules.

Further information:

- The example can be fully investigated online: SmAV with Requirements.
- A short introduction is given in the SpecIF Introduction and Quick Start Guide.

Solution Proposal

Describing a solution proposal goes beyond the scope of this paper. A concept for a system engineering collaboration platform covering use-cases similar to "Model-Exchange" and "Requirements Engineering & System Design" can be found under the following link. The use-case "Verification & Validation" has not been elaborated yet, but it would be realized with the same approach.

- Solution Proposal: System Engineering Collaboration

The solution proposal itself is another example of using SpecIF for system specification: The same methods apply for mechatronic products, just the notations and tools are different. The business processes have been modeled in BPMN using the Camunda Modeler, the application landscapes as well as the information model have been authored in Archimate notation with Archi and the requirements have been collected in Excel. All partial work results have been integrated using the *SpecIF Model Integrator and Editor*. Please note that model-elements with the same title and class have been merged automatically.

Requirements per Use-Case

The requirements stated in Chapter 3 are discussed in the following Table 5.

UC	Requirement	System solution based on SpecIF
MEX	IP Protection	Model content for exchange is limited to selected parts by a suitable model package definition and an export filter. An alternative to exchanging models by file is using a common collaboration platform. Access to model-elements is given based on user identity, roles and permissions on element level.
MEX	Linking of model elements	Being graph-data, relations between any model-elements are an inherent feature of SpecIF.
MEX	Exchange of real model data	Using SpecIF, an empty model structure can be transferred from one authoring tool to another and returned with added details.
MEX	Exchange of (parts of) profile information	It is a feature of the model exporter/importer, which information is included. SpecIF as such can represent also profile data.
MEX	Specification of requirements and expectations	Use-cases, requirements, system interfaces and behavior can be expressed using the SpecIF domain classes for "Model Integration".
MEX	Model exchange on different levels of abstraction	Repeated model exchange is made possible using SpecIF revisions of individual model-elements and relations. Updates on different levels of abstraction can be implemented with suitable export and import filters.
MEX	Exchange minimal set of SysML Diagrams and Elements	Logical consistency in case of partial model exchange is ensured, if the set of SpecIF classes isn't altered by one partner without agreement. SpecIF statement classes can limit certain resource classes for subjects and objects. For example, the class for an IREB:satisfies statement may define that only resources of class FMC:Actor can be used as a subject and only resources of class IREB:Requirement can be an object. In addition, the SpecIF model-element revisions support proper configuration management of a model and its packages ("configuration items").
MEX	Exchange of diagrams	Using SpecIF, original diagrams can be exchanged as resources pointing to an image in any format supported by web-browsers; SVG is preferred.as (SVG carries the same information as XMI-DI, namely explicit geometric information and references to the corresponding model-elements, but in addition it can be directly rendered by web-browsers.) Thus, SpecIF conveys both the visual and the logical aspects of a model.
MEX	Exchange of link information	Being graph-data, relations between any model-elements are an inherent feature of SpecIF.
RQM	Importing requirements into the system model via ReqIF	A bidirectional transformation between SpecIF and ReqIF is available. ReqIF data can be merged with data from other modeling tools. Similarly, requirements in Excel® files can be merged with data from other tools.

UC	Requirement	System solution based on SpecIF
RQM	Use of synchronization plugins / addons	SpecIF data can be imported, exported or synchronized using plugins or addons for any modeling tool. A live synchronization of model data from Enterprise Architect® to Jira® has been implemented using the Kafka event framework.
RQM	Use of a traceability platform	A traceability platform can equally be fed with SpecIF data, if so desired.
RQM	Atomic traces between requirements and model elements	Being graph-data, relations between diagrams, model-elements and between model-elements are an inherent feature of SpecIF.
RQM	Versioning of traces between requirements and model elements	With SpecIF, both resources and statements may have revisions supporting proper configuration management. The configuration management as such is a feature of a tool, though. Rather than marking a trace 'suspicious' when a model element is updated with a new revision and manually confirming those (or not), it may be con-sidered to classify relations as 'floating' or 'static' to avoid such manual confirmation.
RQM	Model-based requirements exchange	Using SpecIF, system models with diagrams, use-cases, model-elements and traces=relations between model-elements, can be exchanged between collaborating companies using the Stakeholder Request Clarification Process (SRC) of prostep iViP. All resources including diagrams and model-elements get the 4 properties per supplier for exchanging status and comments.
RQM	Model-supported requirements exchange	See point model-based requirements exchange in above line.
V&V	Verification and Validation	A set of SpecIF classes can be configured, so test-cases for verification and validation may be included in a system specification and linked to requirements or system components.

Table 5: Discussion of Requirements with focus on SpecIF

Decision Criteria

While it may be very tempting to select an existing tool, the makers of SpecIF believe that a solution for the future must comply with the following criteria. Tool interoperability ensured by an open standard and open access for all market players are indispensable for innovation and flexibility to decide.

Decision Criteria	Motivation	SpecIF Compliance
Data is represented as a **graph**	A system model is a graph by nature. Objects/model-elements are represented as nodes/resources and relations are represented as edges/statements. **Only** solutions based on graph-data are highly scalable and allow performant queries including the relations.	**Yes.** SpecIF provides propositions for at least propositional logic (zeroth-order logic). SpecIF has been successfully mapped to Turtle/RDF for immediate import into graph databases.
Elements have **types/classes**	Types enforce a consistent structure of similar nodes resp. edges, which is essential for efficient modeling and searching. For example, systems can use explicit types to automatically configure/generate databases and user-dialogs including input validation.	**Yes.** SpecIF resources as well as statements may have multiple classes with individual property sets. A property has one of 7 elementary data-types known from XSL. Statement classes may define permissible resource classes as subject and object.
Types/classes can be **configured**	Simple applications like requirements management get along with a simple type-set and more powerful applications like system engineering can get a more refined type-set. In case of growing demands, the type-set can be easily enhanced, while the underlying schema persists. If built properly, the tools continue to work as before, when a type-set is modified.	**Yes.** If desired, a higher level schema can be implemented to enforce a certain type-set.
Information model has a **schema**	Use established technology for • checking and transforming the data • generating software code in many programming languages.	**Yes.** JSON-Schema for SpecIF v1.1 has been released. XML-Schema can be implemented.
Information model is an **open standard**	All system users and vendors must have free access and the opportunity to further develop the standard.	**Yes.** SpecIF is a GfSE standard and standardization by OMG is planned. SpecIF is open-source and has a free Apache 2.0 license even for commercial use.

Table 6: Collection of Decision Criteria and Motivation

ReqIF also complies with all criteria listed above, as well. SpecIF may deal with multiple languages and element-level revisions. For a given language and revision level, SpecIF and ReqIF data can be trans-formed without loss in both directions.

5.1.3 Project MTIP

Introduction

In January 2022 new plugins for model exchange were introduced by the Aerospace Corporation. The Aerospace corporation operates the federally funded research and development center (FFRDC), which has a strong focus on the space enterprise in the US (The Aerospace Corporation, 2022). In April 2022 a team of the Aerospace Corporation presented their approach in one of the INCOSE-LA (Los Angeles) speaker meetings. The initial presentation is public available at the INCOSE_LA chapter (Severson, 2022). The main challenge that was addressed, is the missing support of models that have content outside of the UML standard. This challenge of course applies to SysML.

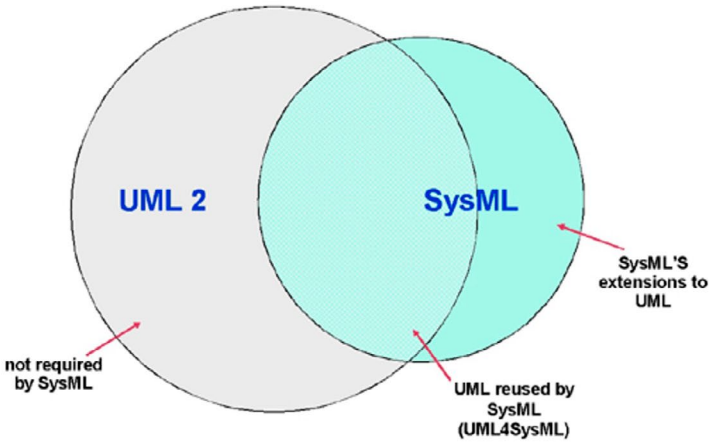


Figure 30: Relation between SysML and UML (OMG SysML, 2022)

Commercial tools often use XMI for model import and export of UML and for that reason XMI is also often seen as the exchange format for SysML. But especially the content of models that uses SysML specific content outside of the intersection of UML and SysML, is in most cases not supported. This was the motivation for Aerospace company to find a solution for SysML model exchange.

Approach

The approach that was suggested is, to use an intermediate format that is mapped towards the different metamodels of the system modeling tools. Therefore, as a first step, the metamodels of Sparx Enterprise Architect and the Cameo Systems Modeler were mapped to a common schema (see Figure 31). This common schema is used for the definition of the exchange format, and is announced to be published in the near future.

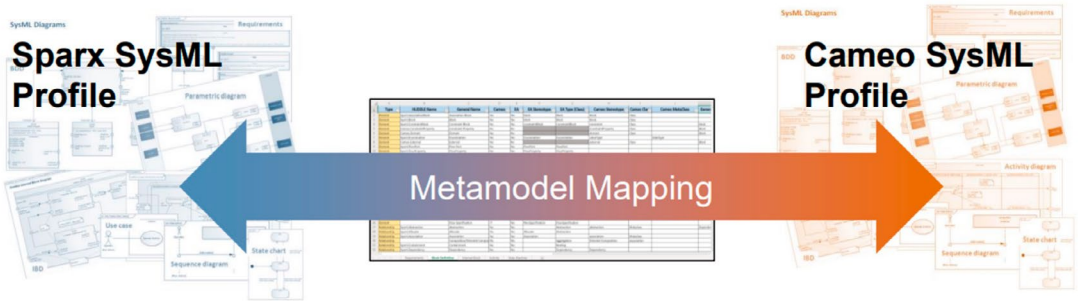


Figure 31: Mapping of metamodels to a common schema (Severson, 2022)

As the actual exchange format, a HUDS (Huddle Unified Data Schema) XML is used, implementing the previously mentioned common schema, derived from the metamodels. This format allows a tool neutral storage of SysML model content, which can be imported later again by using a tool individual mapping schema.

Currently a mapping schema exists for Sparx Enterprise Architect and the Cameo Systems Modeler. In general, this approach is open to be implemented to further system modeling tools like e.g. IBM Rhapsody.

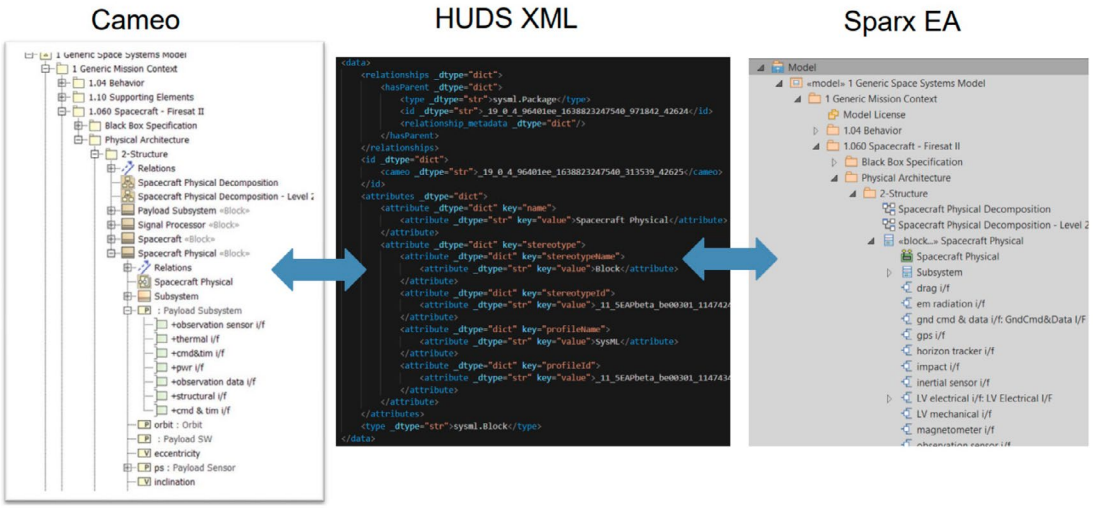


Figure 32: SysML model exchange via HUDS XML (Severson, 2022)

Current implementation

The implementation is compared to other solutions still very new and needs to be evaluated in detail. Nevertheless, the team from the Aerospace Corporation made their current development public available and licensed it under the Apache Commons License 2.0. Two plugins can be downloaded from their GitHub repository, one for Cameo Systems Modeler and one for Sparx Enterprise Architect (The Aerospace Corporation - Github, 2022).

The plugins allow an import and export of model content via the HUDS XML format. The plugins work in a bidirectional way, which means both plugins can import and export models. In fact, this does not mean that it is already possible to use the plugins for a roundtrip model exchange. So, the addressed scenario from the SysML WF to export a model, edit in another modeling tool and import the changes back into the previous version is currently not supported.

The example model of the SysML WF, the eHSUV, does not work out of the box with the plugin and some modifications have to be made to be able to export and import the model. Other examples produced good results in both directions.

Of course, an official benchmark for this plugin would still be necessary to get a detailed view on the capabilities.

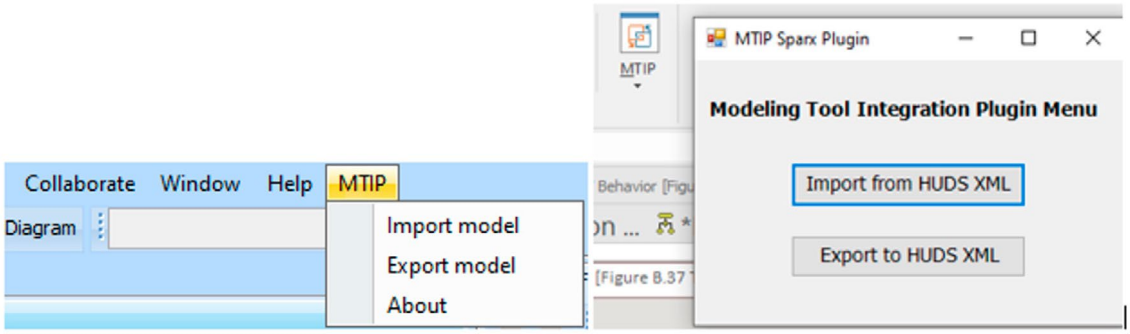


Figure 33: MTIP Plugin for Cameo Systems Modeler and Sparx Enterprise Architect

Regarding the current capabilities, it can be stated that a general support of diagram exchange is existent. This includes the information about which elements and which relations appear on the diagram of each element on the diagram. Not supported at the moment are formatting information like path styling or colors.

The following Figure 34 shows an example of a model exchange via MTIP plugins from Cameo Systems Modeler to Sparx Enterprise Architect. The example model that was used was one of the standard examples (Vehicle Climate Control System) available in Cameo. The image shows a Block Definition Diagram (bdd) and it can be seen, that the diagram still contains all the information and the position of the blocks, but the path layout got changed and would need some correction inside Enterprise Architect.

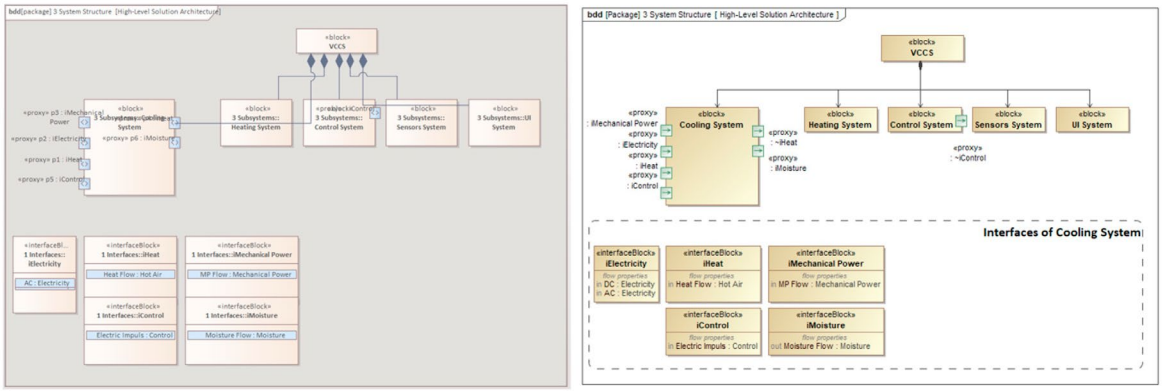


Figure 34: Block Definition Diagram transferred via MTIP from Cameo to EA

Known gaps

The Project MTIP approach and implementation left a good impression inside the WF and IF. Nevertheless, there are also some points that already revealed during the first analysis.

As already mentioned the solution is so far not capable of roundtrip scenarios, which is one of the essential needs of the WF for a future model exchange solution. But it was already stated by some of the vendors that the effort for the extension of the current source code regarding the round-trip capability would be manageable.

In addition, the plugins currently do not give the capability to export and import only partial models, which means that which the current version only full models can be exchanged. This might also lead to some problems regarding IP protection.

Last but not least the current implementation only supports Cameo Systems Modeler and Sparx Enterprise Architect within specific version. E.g. the Cameo plugin only works with version 19 and is so far not supported inside the current version 2021x. For Enterprise Architect the plugins are compatible with version 14.x and 15.x

Outlook

With the Project MTIP implementation there would be a good starting point for a future standard for model exchange of SysML models. As already mentioned there would be still some gaps that need to be addressed and solved to have a solution that is closer to the needs addressed by the WF.

Some of the vendors inside the IF already showed interest on working and contributing to the implementation, but a final decision has in most cases not be taken.

The team of the Aerospace Corporation already gave an outlook on their plans regarding the further development and stated, that they will work on the support of additional metamodels (i.e. UAF, DoDAF, UML) and of course they also count on feedback and testing through a wider userbase to continue and improve with their implementation.

Already announced was also that the Project MTIP will join the OpenMBEE community and also the publication of their HUDS XML schema is planned in the near future.

5.1.4 Tool-to-Tool integration

The previous chapters show, that a standard for SysML model exchange, that fulfills all the needs addressed by the WF is still not yet available. All current solutions and approaches only fulfill partially the needs or support only specific tool combinations.

Due to the missing standard, some vendors and third-party companies implemented bi-directional tool integrations for specific tool combinations. These tool integrations often use proprietary formats that cannot be used for other tool combinations and are provided as commercial addons or plugins. This might lead to the situation that a supplier would need many different solutions for his different collaboration scenarios. In addition, the compatibility of these addons needs to be considered. Version upgrade of one of the tools, directly leads to a version upgrade of the interface.

That's why the specific integrations are currently not the preferred solution by the WF members and a common standard, maybe based on SpecIF or Project MTIP would be preferred.

5.2 Demonstrators presented by IF vendors

This chapter gives an overview on the different demonstrators and how they are linked to the described scenarios. Not all demonstrators refer to the topic of model exchange, some of them also address the use cases collected in the RE and V&V work package.

All demonstrators were presented towards the SysML WF and feedback was collected (feedback is summarized in Chapter 6).

5.2.1 LieberLieber Demonstrator

LieberLieber used their LemonTree software to show a concept for the exchange scenario described in WP4. The solution showed by LieberLieber brings two interesting aspects into the discussion of the Model Exchange Use Case. The first point is the usage of a merge approach instead of a direct exchange of complete models and the second point is the usage of OpenMBEE (Open Model Based Engineering Environment).

OpenMBEE is an open source collaborative engineering system and offers different MDKs (Model Development Kits) for exporting models out of different tools by implementing the DocGen language (OpenMBEE, 2022). These exports can then be synced into the OpenMBEE MMS (Model Management System) (see https://www.openmbee.org/index.html).

LieberLieber took the approach of OpenMBEE MDK to be able to export models out of the authoring systems (e.g. Cameo) and make the information readable for their LemonTree software. The following Figure 35 shows the OpenMBEE Cameo MDK plugin.

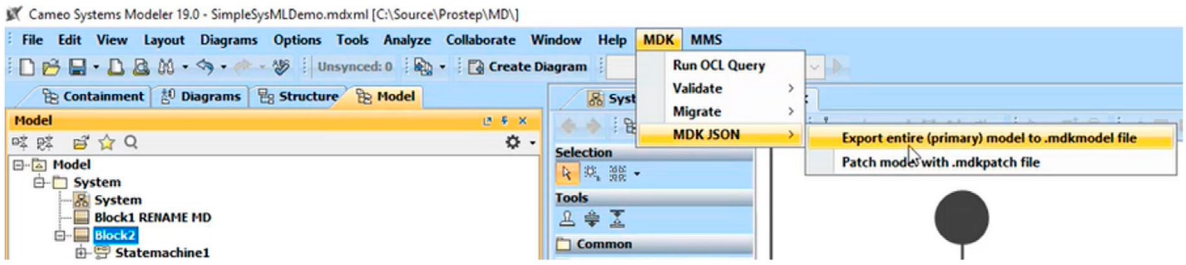


Figure 35: OpenMBEE MDK for Cameo

Using the plugin, it is possible to create a file in the "mdkmodel" format, which can be read by LieberLieber's LemonTree software. This allows in a following step a merge between e.g. an Enterprise Architect project and the exported MDK model, that was created out of Cameo.

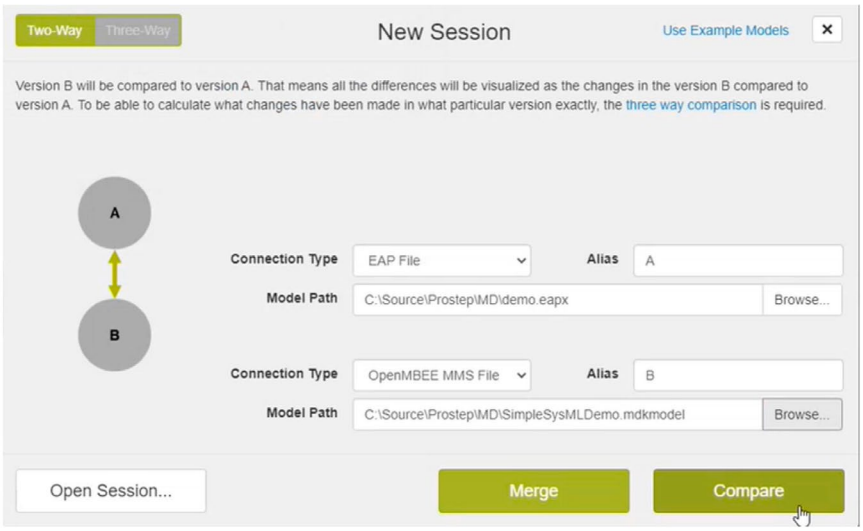


Figure 36: Selection of files to be merged inside LemonTree

With this capability LemonTree makes a collaboration between two companies, that work with different system modeling tools, possible. Of course, a clear process has to be defined for who is responsible for the merge activities and which model defines the main structure.

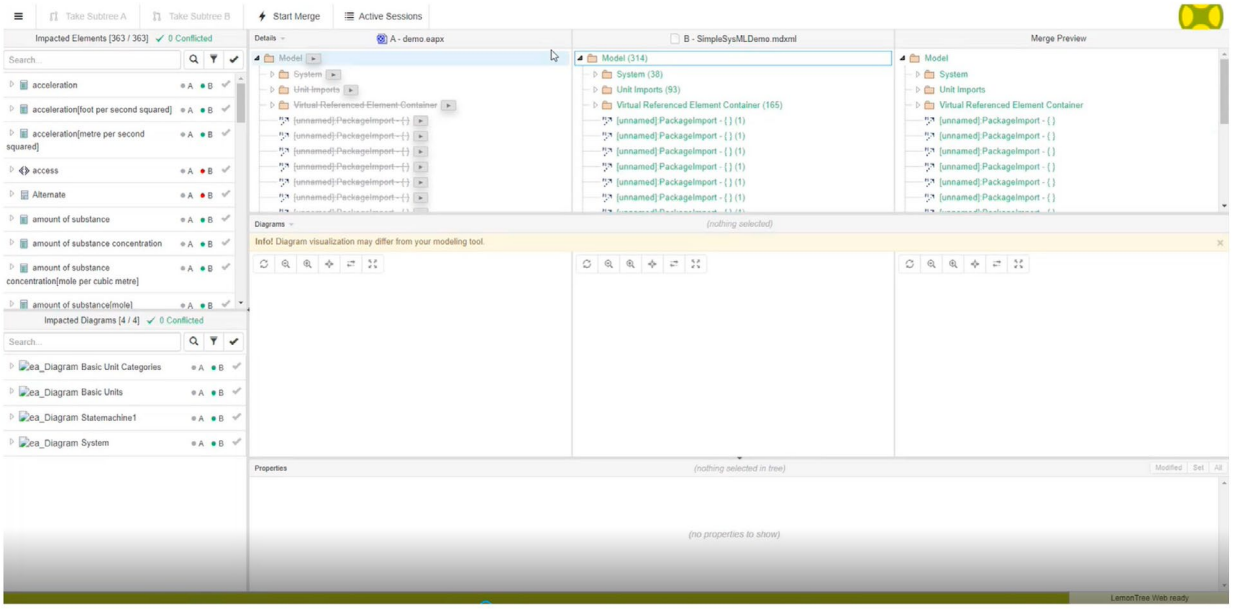


Figure 37: LemonTree merging an EA and OpenMBEE MDK model

LieberLieber also showed in a former demonstration, that they could also merge Cameo and EA files by using XMI instead of OpenMBEE. For LieberLieber the approach of using OpenMBEE seems to be more promising than using XMI for future use cases, why they stopped the further development of the XMI approach at the moment.

5.2.2 MID Demonstrator

MID focused with their demonstrator on the WP4 and WP6. In fact, MID presented two demonstrators, one for each of the two work packages.

For the WP4 "Model Exchange" MID presented their solution for exchanging system models based on their "Open MBSE Data Package". For different system modeling tools, they can already provide adapters, that allow reading and writing based on their "Open MBSE Data Package". The basic concepts can be seen in the following figure.

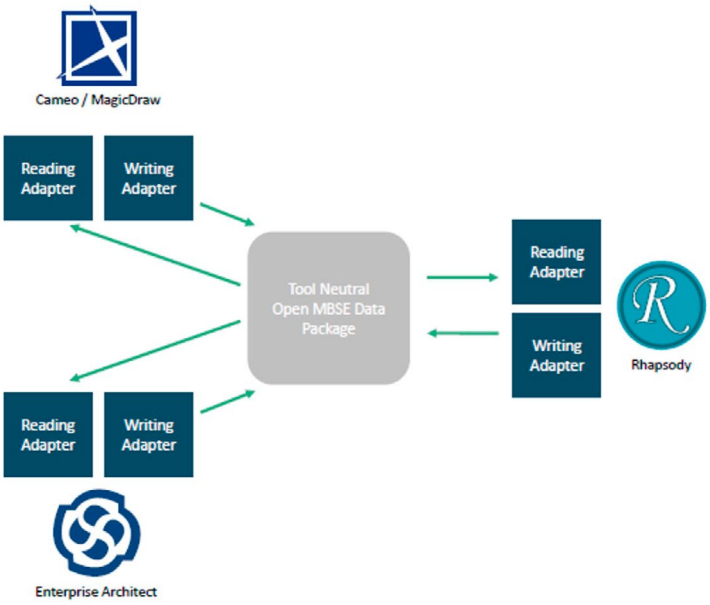


Figure 38: MID exchange concept based on their Open MBSE Data Package, Source: MID

This concept could be extended by adding further reading and writing adapters for other system modeling tools. The tool neutral "Open MBSE Data Package" contains the following artefacts:

- JSON
  - nodes and their properties
  - ownership tree
  - Relationships
- Manifest with the meta model
- SVGs
  - Visual Shapes 1:1
  - Clickmaps integrated into SVGs + UUIDs
- Position of shapes and relations
- Original Model

With this approach it is possible to also support the exchange of diagrams between different system modeling tools, which was a central requirement addressed by the SysML WF. The following figure shows an extract of the MBSE data package that was written out of Cameo for the eHSUV example. It can be seen, that it is actually a ZIP Container which stores the relevant system data based on the combination of JSON and SVG files.

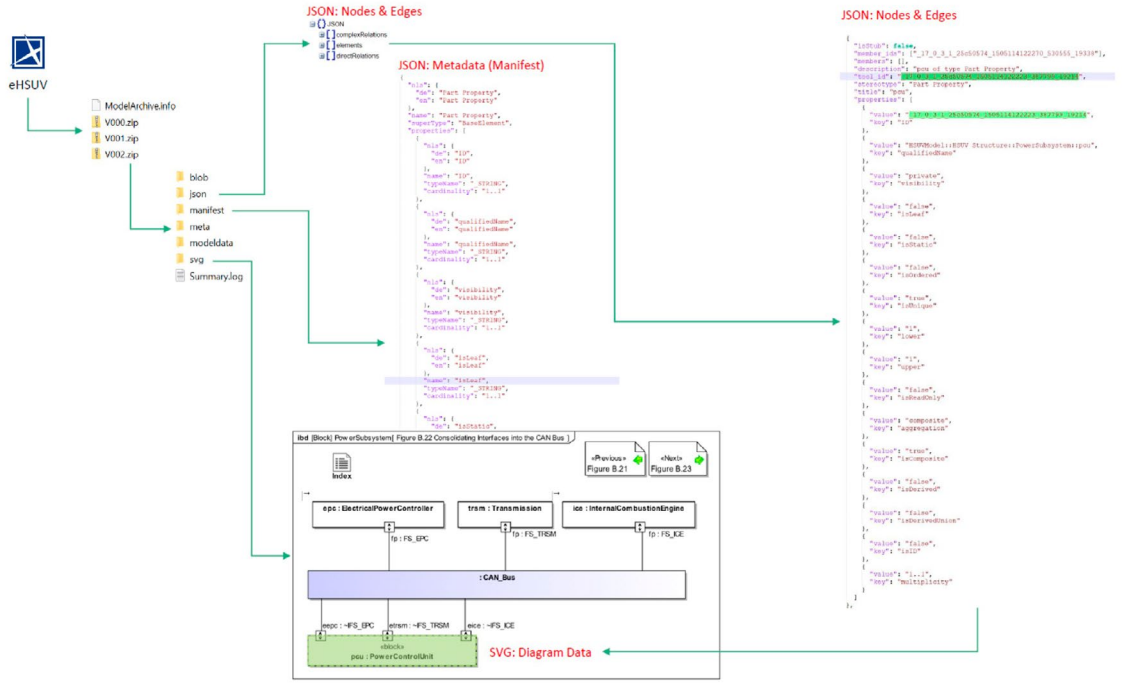


Figure 39: Extract of the Open MBSE Data Package for the eHSUV example

In addition to the direct exchange between different system modeling tools, MID also offers a collaboration platform called smartfacts, which can also read the MBSE Data Package. Using smartfacts several functionalities are offered that go beyond the pure exchange of models:

- Consume, navigate investigate a SysML model manually without any tool
- Do a Version Diff
- Comment and review the model
- Link model elements with other tools

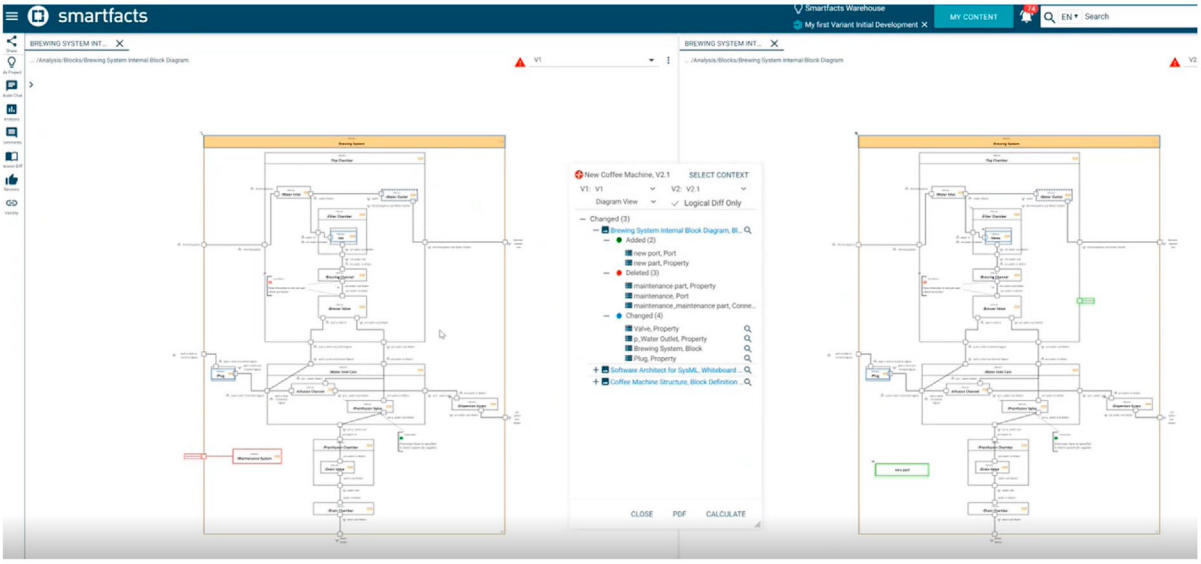


Figure 40: Showing a version diff inside smartfacts

Regarding the demonstrator for WP4 MID also gave an outlook for the next steps, that could be implemented:

- Currently the transformation rules are static. It is planned to separate the rules into a JSON based rule profile, which will allow an obfuscation of the published model content
- Create log files for the model transformation
- Profiles for the receiving tools must be setup manually at the moment, it is planned to have an automatized profile transformation in the future, transforming a source into a target profile

The second demonstrator that was shown with reference to the WP6 is based on the already mentioned collaboration platform smartfacts from MID. The collaboration platform allows the linking of requirements and system artefacts across tool borders.

Therefore, the different models and specifications are published into the smartfacts platform using the already mentioned Open MBSE data package in combination with the OSLC standard, see following figure.

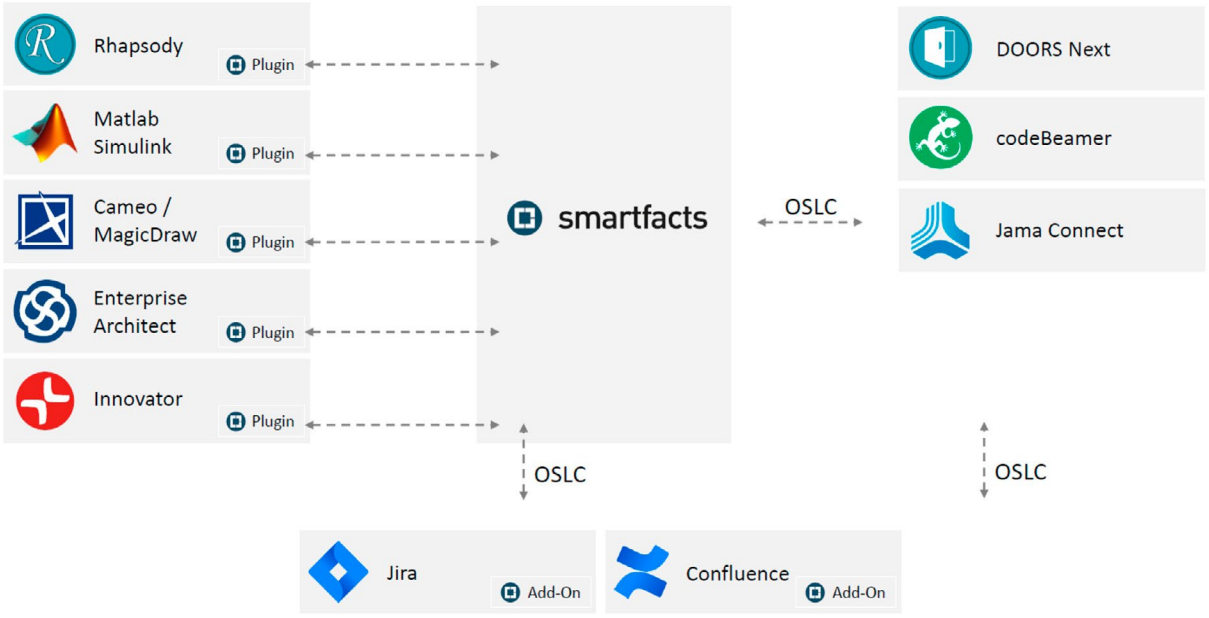


Figure 41: Concept of MID collaboration platform, Source: MID

Emphasized by MID was also the importance of a global configuration. A global configuration concept allows the definition of a specific context, that defines specific versions (local configuration) of each artefact which belongs to the global configuration. This is necessary to also give the links between the different artefacts (e.g. a requirement and a system element) a context.

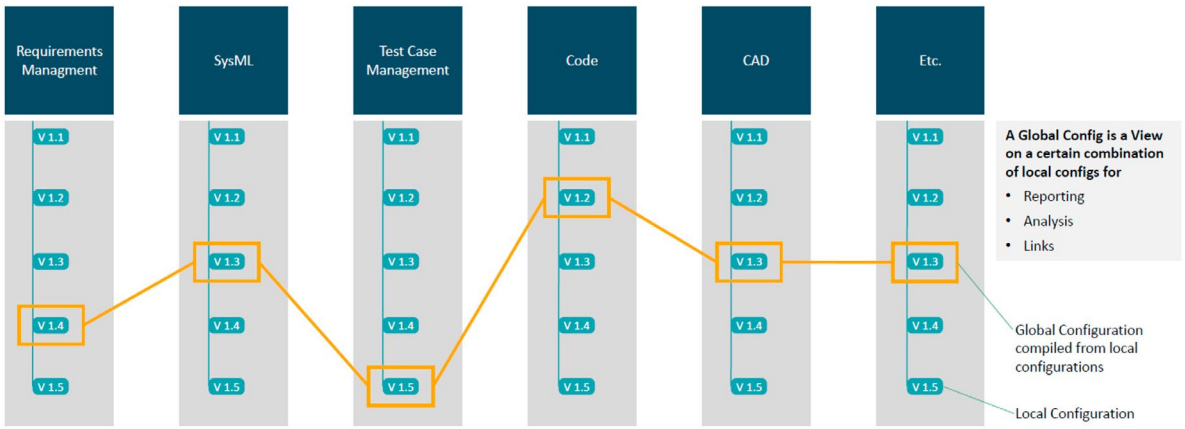


Figure 42: Global configuration concept, Source: MID

For the creation of links, MID offers different options. On the one hand links can be created inside the smartfacts platform, which also allows the navigation through the models and specifications and on the other hand it is also possible to use tool plugins, to create links directly inside the authoring tools, e.g. in Cameo using the smartfacts plugin (based on delegated UI concept), showed in the following figure.

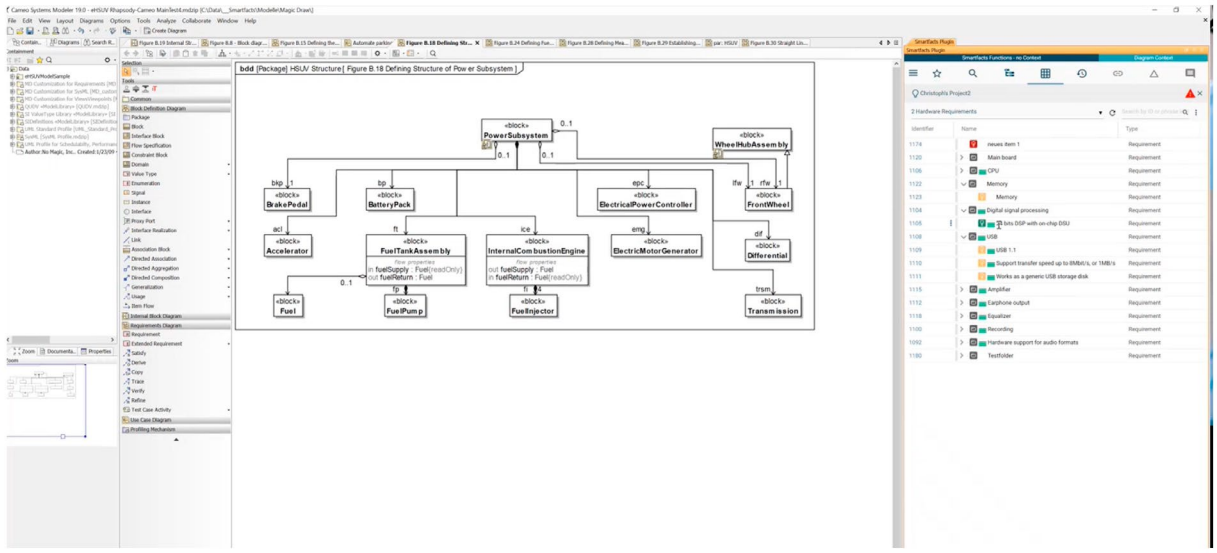


Figure 43: Smartfacts plugin for Cameo

In general, both approaches support two main views:

- Showing full models and specification for creating links
- Showing only linked artefacts defined as part of a global configuration (context) for analysis purpose

5.2.3 Siemens Industry Software Demonstrator

The Siemens Industry Software demonstrator focusses on the WP7 Verification and Validation. It shows an approach of using a holistic PLM system as single source of truth, which allows the traceability across the different artefacts that are relevant for a verification and validation process. The following figure gives an overview on the integrated MBSE approach of Siemens Industry Software.

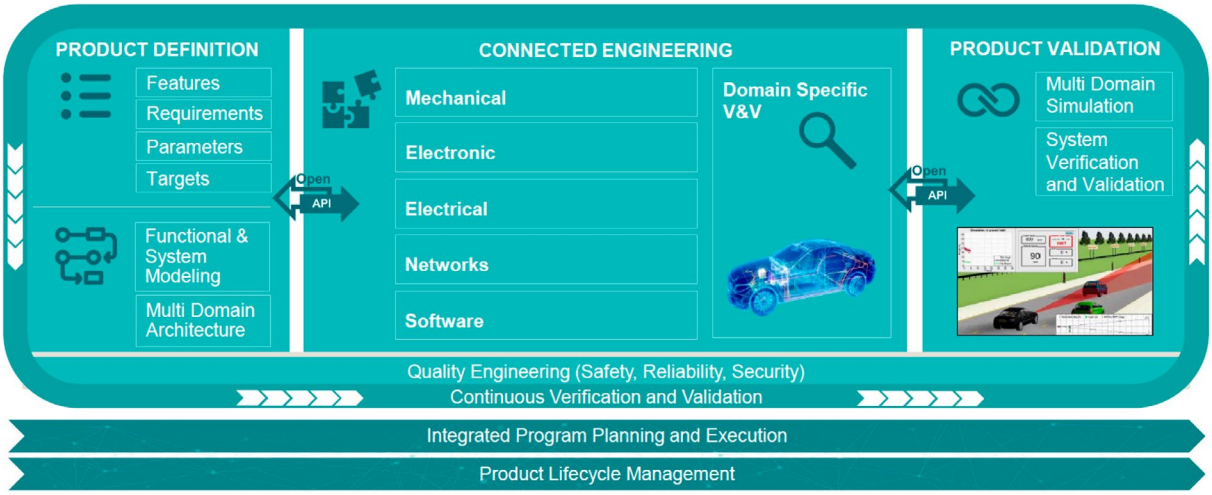


Figure 44: Integrated MBSE Approach of Siemens Industry Software

The demonstrator is based on the PLM System Teamcenter Active Workspace. As a starting point a system model and a requirement specification are stored and versioned in Teamcenter Active Workspace already. A part of the system model, that was originally created inside the Systems Modeling Workbench can be seen in the following figure:

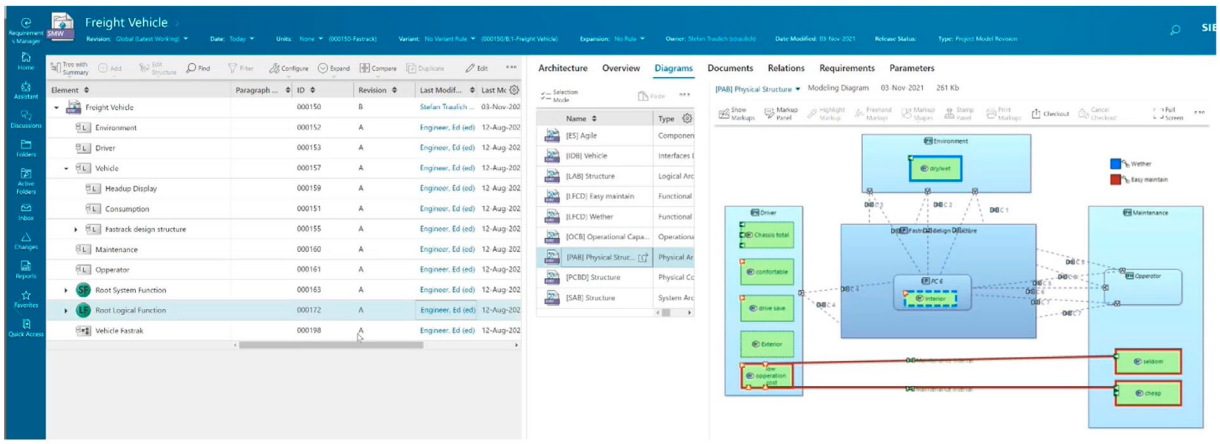


Figure 45: System model managed in Teamcenter Active Workspace

For the definition of the requirements that are addressed towards the system, the Active Workspace requirement environment is used. Inside this environment it is possible to derive parameter objects out of the requirement text. These parameters can be used later in different authoring systems e.g. the CAD model or for simulation models. The following figure show the creation of parameters as part of a requirement.

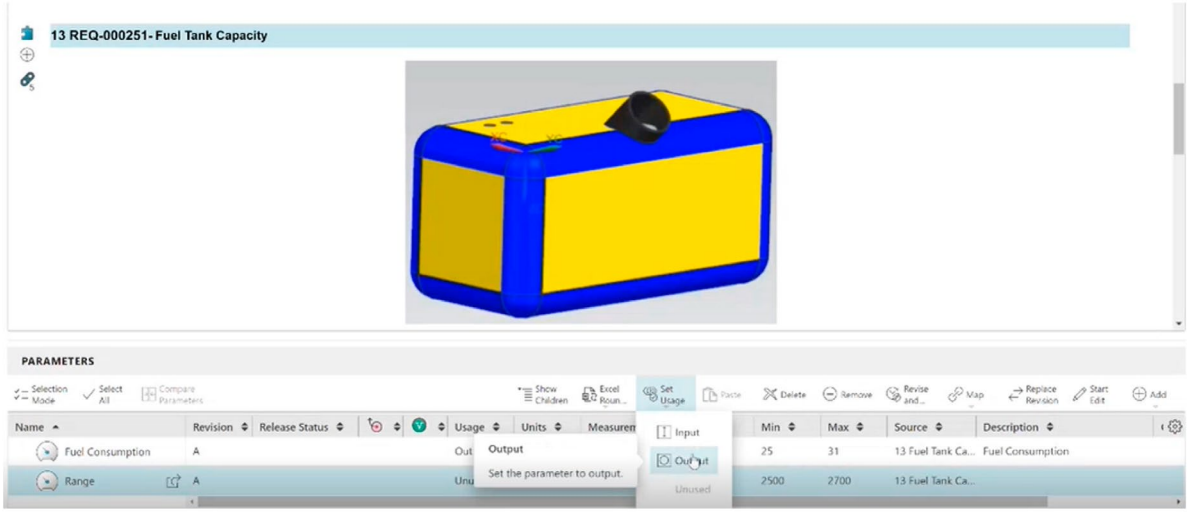


Figure 46: Parameter definition under a requirement inside Active Workspace

As well as the requirements, also the test specification including the verification requests are created and managed inside the platform. For the creation of a verification request all linked information is already collected and prepared by the platform by evaluating the traces between the different artefacts. So, a verification request already contains information about e.g. linked requirements, parameters and functions. Based on the verification request, specific test or simulation request can be created that can also be executed out of Active Workspace by handing over parameters and related models to the specific tools. Inside the demonstrator it was shown by using the MATLAB integration, which was used to calculate the fuel consumption based on the parameters that have been defined in advance inside Active Workspace.

As a further use case the integration of CAD tools (in this case Siemens NX) has been shown, that allows the exchange of requirements and parameters directly with the CAD environment, as shown in the following figure.

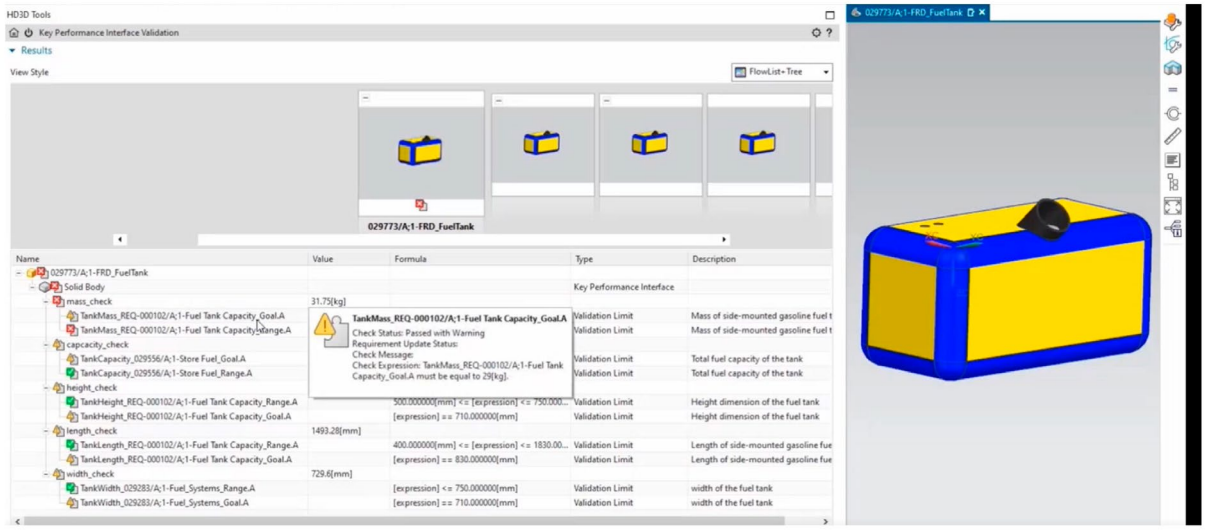


Figure 47: Integration of requirements and verification information into the CAD environment

5.2.4 AVL/CONWEAVER/T-Systems Demonstrator

An example of how an Implementor Forum can encourage the collaboration between the vendors, is the example of the AVL List (in short AVL), Conweaver and T-Systems demonstrator. The demonstrator shown by the three companies is allocated to the WP7 and partially to WP6.

All three companies took one of their software products and defined / developed interfaces to demonstrate a continuous workflow. In particular the following software products were part of the demonstrator:

- T-Systems PDM Webconnector
- Conweaver Linksphere
- Model.CONNECT™ by AVL

The architecture of the demonstrator can be seen below:

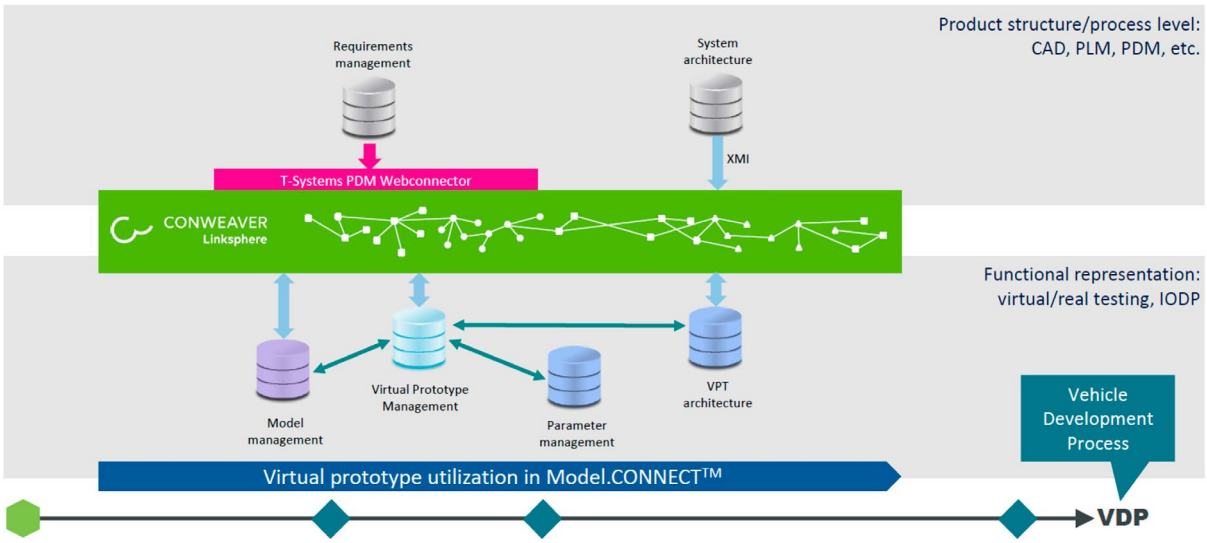


Figure 48: Demonstrator Architecture, Source: SysML WF/IF hand-over meeting

The requirements of the WP7 that are addressed in particular are:

- Atomic traces among artifacts in the area of V&V
- Relations among parameters
- Link to external instances
- Linking of attributes from external sources

Each of the three software solutions takes a particular task inside the demonstrator using their different strengths:

- The PDM Webconnector from T-Systems is used to collect and read data e.g. the system model, requirement specifications or test cases from different authoring tools.
- Conweaver Linksphere takes the collected data and creates a knowledge graph, which allows the link creation between the different data artefacts. Out of this linked data a package can be created that describes a virtual prototype of the system under development. The package is created using the Modelica standard SSP (SSP Standard, 2022).
- The co-simulation platform Model.CONNECT™ finally uses this package, created by the Linksphere platform, and can execute several simulation runs. The results of the simulation runs are transferred back into Linksphere, where the results will be linked to the existing data of the virtual prototype, particularly the requirements.

The following figure shows the Linksphere user interface, where on the left side a part of the system model describing the functionality and interfaces can be seen and on the right side a part of the knowledge graph, showing the traceability between the different artifacts.

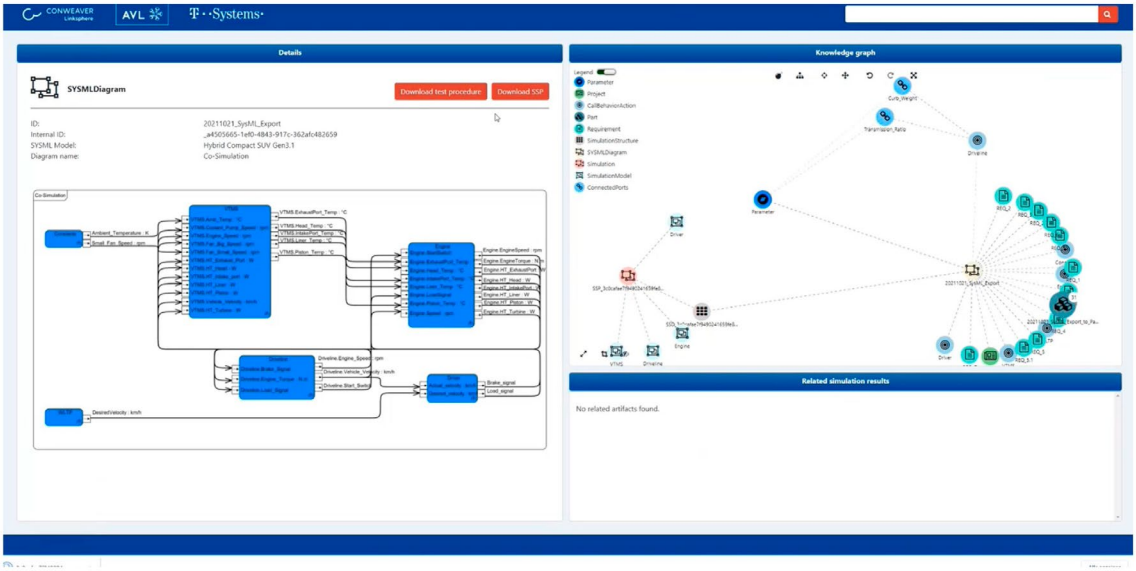


Figure 49: Conweaver Linksphere user interface showing the linked artifacts

The demonstrator showed concepts for integrating and tracing V&V artefacts, in specific:

- Harvesting data sources
- Collecting & linking artefacts
- Serving data to the V&V process

As a next step, the companies plan to enhance their demonstrator by integrating additional data sources, like test management or CAD systems. The vision of the companies is to show an automized roundtrip, that demonstrates a closed loop scenario starting from e.g. a change in the design (CAD model), which leads to updated parameters and simulation models and will lead to updated simulation results. The updated results will be transferred back to the traceability platform, which allows a review of the virtual prototype at any time with the current valid parameters and results, to check the fulfillment of the linked requirements.

The following figure shows the extended architecture, that shall be realized in the next phase.

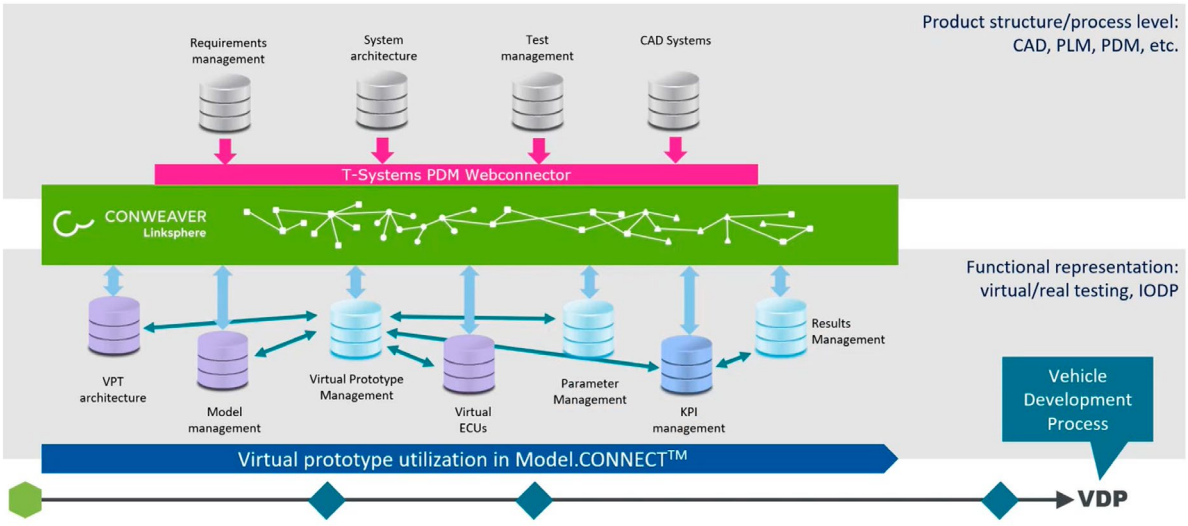


Figure 50: Future demonstrator architecture, Source: SysML WF/IF hand-over meeting

5.2.5 AVL/Dassault Systèmes Demonstrator

The last demonstrator is a collaboration between two companies. In this case Dassault Systèmes and AVL presented an approach of bringing system and simulation models together by using already existing standards. The demonstrator is allocated to the WP7, but in contrast to the previous demonstrator the focus is not on linking data from different sources to describe a virtual prototype, it is more a deep dive into the hand-over between system model and simulation model.

In general, there are already existing solutions on the market for integrating system and simulation models, but in most cases these solutions use specific point-to-point connectors to enable the exchange between the models. The goal of the Dassault Systèmes / AVL demonstrator is, to use standard formats as far as possible.

The basic concept is described in the following figure:

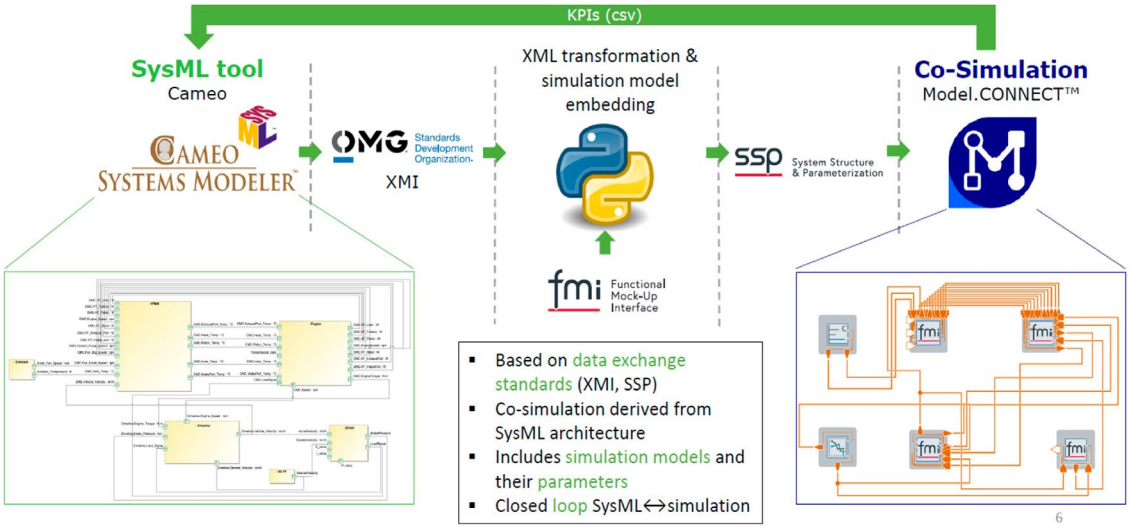


Figure 51: Process description of AVL/Dassault Systèmes Demonstrator

The figure already gives an overview on the used standards that are relevant for the demonstrator. In particular these are:

- XMI
- SSP
- FMI

The tools that are used as part of the demonstrator are:

- Cameo Systems Modeler
- Model.CONNECT™
- Additional Python Scripts

For the use case demonstration, the example of the fuel consumption analysis based on the WLTP was used. As shown in the following figure, the system model describes the basic architecture and its components based on SysML. In addition, the system model allows also referencing simulation models, the configuration of parameter values and of course the link to the requirements.

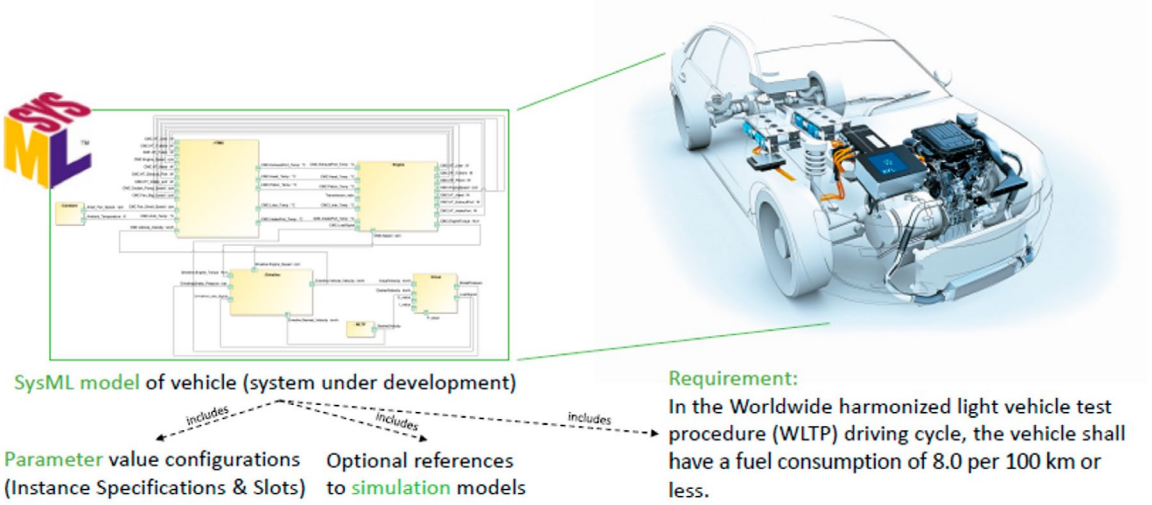
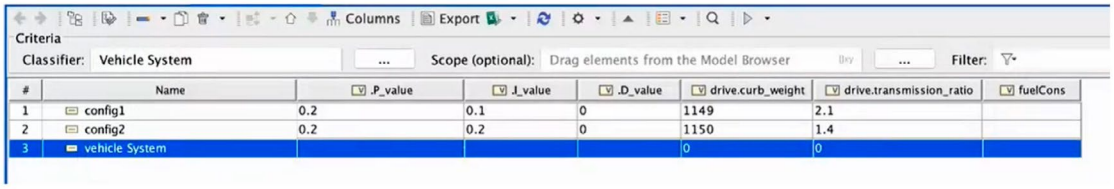


Figure 52: Overview system model content

Inside Cameo the mechanism of instancing system components is used to create different configurations of the vehicle under development, that shall be analyzed later in the process. For each instance it is possible to edit specific parameters. These instances could be configured by e.g. using a product line engineering approach or by just creating some specific instances to test some extreme cases. The following figure shows a spreadsheet view for the configuration of the different instances in Cameo.



#	Name	P_value	J_value	D_value	drive.curb_weight	drive.transmission_ratio	fuelCons
1	config1	0.2	0.1	0	1149	2.1	
2	config2	0.2	0.2	0	1150	1.4	
3	vehicle System				0	0	

Figure 53: Definition of instances / configurations in Cameo

The whole system model including the information about system structure, linked simulation models, instances and requirements is saved as a Cameo project file in the “mdzip”-format.

Based on the “mdzip”-file a python script is used to extract all needed information for the upcoming simulation and transform it into the SSP format. For each configuration a single SSP file is created, that contains in this case an SSD file describing the system architecture and an SSV file containing the parameters for each configuration.

The SSP file can then be imported to Model.CONNECT™. Inside this co-simulation platform, the imported SSP file is linked to a test procedure, to have an executable virtual prototype. In this example the WLTP was used as definition of the test procedure, see Figure 54.

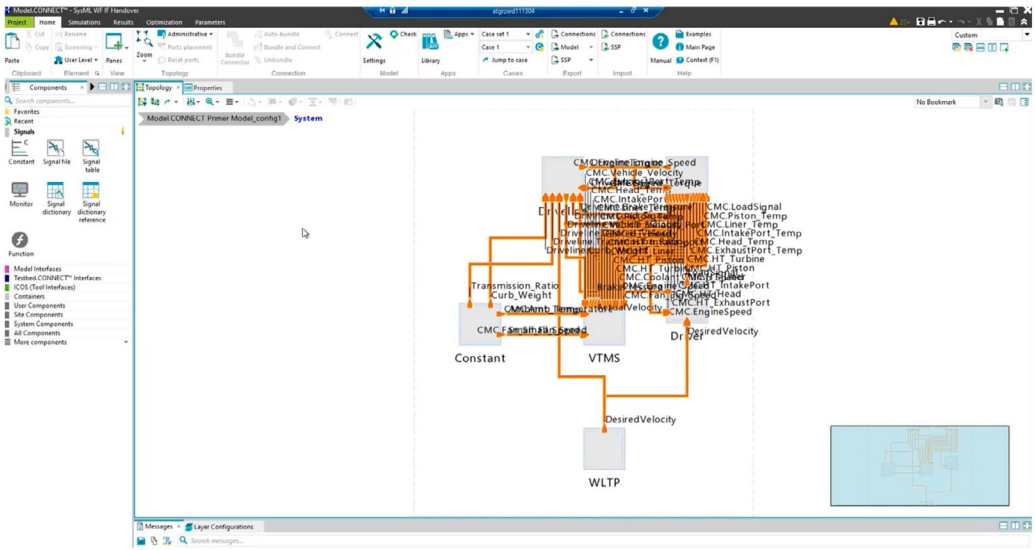


Figure 54: Simulation model imported using SSP in Model.CONNECT™

After all configurations have been calculated with this approach, all results are stored in a CSV file that can be synchronized back into the system model for further analysis. The calculated values for the fuel consumption can be used inside the system model for performing verification activities. Therefore, the different configurations are applied to the predefined set of requirements to verify if all requirements are fulfilled by the specific configuration (see following figure).

Requirement Verification: ☐ Pass ☐ Fail

#	Id	Name	Text	Bounds	Property	Margin	Value
1	Req1	Req1 Fuel consumption	In the Worldwide harmonized light vehicle test procedure (WLTP) driving cycle, the vehicle shall have a fuel consumption no more than 8.0 l per 100 km.	<=8	fuelCons : Real	-2.62	10.62

Figure 55: Verifying requirements based on calculated values

6 Feedback on Demonstrators

6.1 Generic Feedback

Beginning of November 2021, the vendors presented their demonstrators towards the SysML WF members with the request for feedback. Based on the handover meeting and the recorded sessions, the SysML WF provided beginning of December the following general feedback:

- he way of the demonstration, including live interaction inside the demonstrator accompanied by a document / presentation describing the demonstrators, helps to understand the demonstrators and also gives the chance to review the demonstrator again afterwards.
- Some approaches showed more an in-house collaboration rather than a cross-company collaboration.
- For some demonstrator information was missing to understand the concept / idea behind the solution.

The focus for the SysML WF lies on the exchange of SysML models across company boundaries. This includes the whole content of the system model and ideally the linked objects in specific requirements and V&V artefacts. With regards to the demonstrators the following specific points were addressed:

- Diagrams:
  - The exchange of diagrams is necessary because a lot of information is also transferred by using the visualization inside the diagram.
  - The exchange of diagrams has only partially been demonstrated.
  - It is not an identical representation necessary, but it should be close to the original diagram to get the information that the author put inside the graphical representation.
- Openness:
  - The used standards are only partially open.
  - The extension towards other (modeling) software was not always clear / recognizable.
- General approach:
  - The demonstrators showed different approaches and have to be distinguished between real data exchange and linking of data (without exchange).

Open questions that stayed after the demonstration were:

- What are the current limitations of the existing solutions?
  - Due to language conditions
  - Due to technical, implementation constraints
- Does the solution require certain (style/modeling constraints)?
  - Can a universal understanding be found that can be released as a recommendation?
- What are the underlying concepts of the demonstrators (methodologically)?
  - On which standards and which method is the demonstrator solution based?
- Where is more specification needed from the WF?
  - E.g. specification of workflow steps, …
- What is the vendor's view on currently existing / planned standards?
  - E.g. SpecIF, OpenMBEE, MTIP, XMI, …
- Can a common understanding be reached to drive a standard (analogous to ReqIF)

Besides the generic feedback, the WF also provided demonstrator specific feedback with is documented in the following chapter.

6.2 Demonstrator Specific Feedback

The specific feedback on the demonstrators is as heterogeneous as the challenges addressed and requirements that were at least partially met. In order to harmonize the feedback, the well-known V-Model is used. This approach allows a general overview of the main challenges in SysML collaboration, as an individual comparison of the demonstrators is difficult due to the given diversity. For the WP4 (Model exchange) and WP7 (SysML – V&V) respectively WP6 (SysML – RQM) demonstrators three main aspects of the V-Model are of crucial importance:

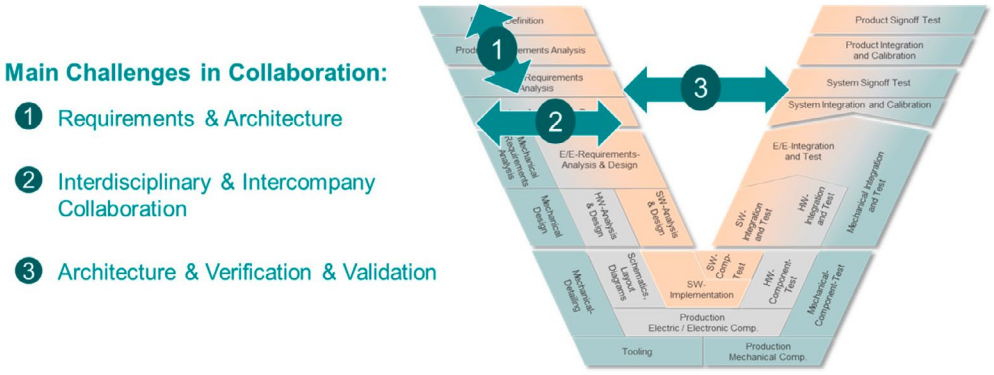
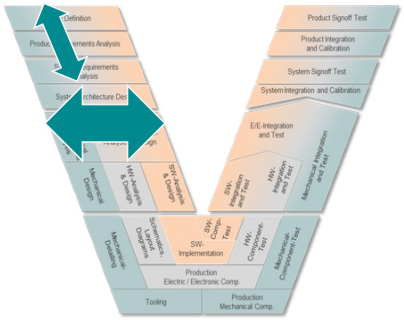


Figure 56: Main Challenges in collaboration according to V-Model

6.2.1 LieberLieber Demonstrator

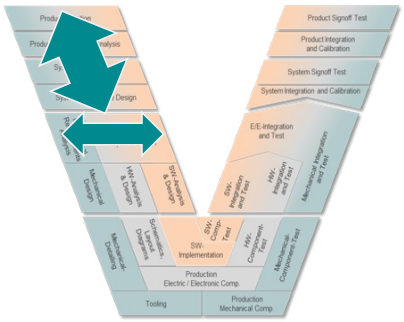


Within the V-Model, the LieberLieber Demonstrator for WP 4 (Model Exchange) is located on the upper-left corner, as it can process requirements in the open source OpenMBEE MDK format. By using this open format, LieberLieber's Demonstrator enables cross-company collaboration, which is one of SysML WF's goals. Concerning the V-Model of Figure 56 LieberLieber's Demonstrator is able to build an interdisciplinary bridge between different domains, so the main challenge of intercompany collaboration is successfully addressed.

Due to the fact that the LieberLieber Demonstrator fulfills the requirement to process the relevant SysML artifacts and elements, the exchange of partial SysML models and real data is possible in a separate merge or diff concept. Thus, this demonstrator enables a way of working where a model can be moved across different tools. From SysML WF perspective this aspect is seen as a promising way to follow.

A gap that is seen by the SysML WF at the moment, is the missing support for diagrams. Currently no diagrams can be exported by using the OpenMBEE MDK. This point is acknowledged by LieberLieber and they are thinking about an extended solution which might support diagrams in the future.

6.2.2 MID Demonstrator



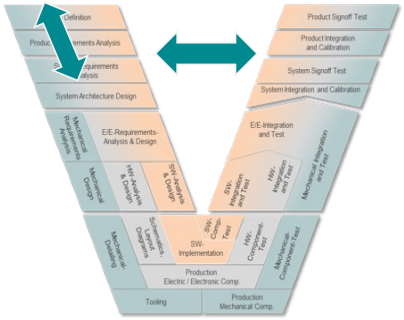
MID has presented one demonstrator each for WP4 (Model Exchange) and WP6 (SysML – RQM).
Within the framework of MID's Open MBSE Data Package, the WP4 demonstrator can process relevant SysML artifacts and elements. Diagrams as well as standards like OSLC or formats like ReqIF are supported. The transfer or linking of data (e.g. requirements) is possible. However, MID uses an internal data format (Open MBSE Data package) to exchange SysML models. As a consequence, this means that no completely open exchange is possible. In addition, model obfuscation (e.g., for intellectual property protection) is not yet supported.

The demonstrator designed for WP6 (SysML – RQM) showed how to link requirements and system artefacts across tool borders, which fulfill requirements such as the model-based requirements exchange. For that reason, the strengths are located on the upper-left side of the V-Model.

MID could show with both demonstrators, that many of the requirements addressed by the SysML WF can already be fulfilled, by combining the concept of a neutral exchange format together with a collaboration platform that allows the creation of traceability across tool borders using standard technologies like OSLC.

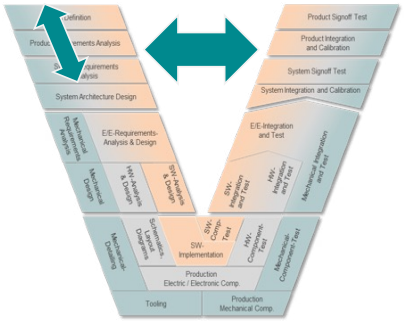
In contrast to the previous demonstrators described the following demonstrators are allocated to the WP 7 (SysML – V&V), which focuses on verification and validation. As WP 7 is an extension of WP 6 (SysML - RQM), the V-Model is addressed at two points according to Figure 56: Upper-left side and between the two sides of the V-Model.

6.2.3 Siemens Industry Software Demonstrator



The Siemens demonstrator handles the WP 7 (SysML – V&V) and is therefore located between the two sides of the V-Model to show a traceability between requirements and test cases. It is able as well to perform coverage analysis. The demonstrator showed the potential of having all relevant artefacts linked to each other and having them available at all time. Based on the available data, it is possible to monitor and track the process of the development and to automate workflows for creating and handling data across the whole verification process. A good representation of the results is given by a dashboard view. The requirement of showing atomic traces between requirements and model elements and versioning of traces is fulfilled within Siemens environment. However, the demonstrator has no focus on SysML and works in a dedicated Siemens environment. Because of this one-tool solution, intercompany collaboration with SysML models is difficult to handle.

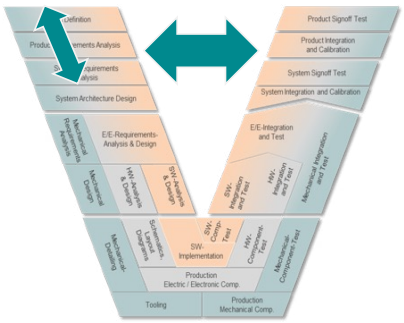
6.2.4 AVL/CONWEAVER/T-Systems Demonstrator



Within the demonstrator of T-Systems, Conweaver and AVL it is possible to connect data from different tools and to link SysML artifacts, requirements, system and simulation models as well as parameters. This satisfies the requirement to show atomic traces between requirements and model elements and to version these traces. Doing so the demonstrator could be developed as a central data hub for linking information from top left side of V-Model to top right side of V Model. Hence the strengths of this demonstrator are located between the both sides of the V-Model.

To achieve this, several steps still need to be taken, like the completion of the SysML Connector for the PDM Webconnector of T-Systems. Another improvement could be the integration of test cases.

6.2.5 AVL/Dassault Systèmes Demonstrator



The demonstrator showed a closed loop approach for the verification and optimization based on a system model by using existing standards. For this purpose, the conversion from XMI format to SSP format is very useful here. As the focus of this demonstrator is to enable a consistent exchange between system model and simulation model, the strengths are also between the two sides of the V-Model. An open point that could be addressed in the future is the automatization of this workflow. Currently most steps demand a manual transfer of the information between the two tools. As already mentioned in the T-System/Conweaver/AVL demonstrator, also for this demonstrator a further enhancement could be the integration of test cases.

6.2.6 Summary

Table 7 below provides an overview of the demonstrators' key strengths and weaknesses.

Demonstrator	Strengths	Weaknesses
LieberLieber (WP4 Model Exchange)	• Cross-company collaboration • Open-source format	• Missing diagrams
MID (WP4 Model Exchange + WP 6 SysML – RQM)	• Cross-company collaboration • Traceability of Requirements	• MID specific format, no open access
Siemens (WP 7 SysML – V&V)	• Atomic traces	• No model-based exchange between different tools possible
T-Systems/Conweaver/AVL (WP 7 SysML – V&V)	• Traceability of model elements • V&V	• Missing test cases
Dassault/AVL (WP 7 SysML – V&V)	• Closed loop of verification	• Manual workflow

Table 7: Summary of feedback on demonstrators

7 Recommendations and Next Steps

7.1 Recommendations

Based on the previously described contents and work results of the SysML WF/IF, the following recommendations are made to Collaboration Workflow, to Tool Vendors and to Committees.

7.1.1 Recommendations to Collaboration Workflow

The collaboration scenarios between different engineering parties are described in chapter 3, where the major use case 3.1 describes the model exchange between the OEM and the supplier. Therefore, some good practices were investigated during the exchange process of a SysML model between an OEM and a supplier.

In addition to models and used tools the implemented method has a huge impact on the exchange process. In the automotive industry different engineering methods are used like SPES, SysMod, SmartGrid, Harmony SE, … The methods help and guide the architect during the engineering process. Usually, a meta model defines the required engineering artefacts and their relations including the used SysML elements.

Mostly exchange partners use different methods and may have different SysML elements defined for the same engineering artefacts. This can lead to misunderstanding of the transferred model content, because SysML is only the language which defines a set of model elements and how they can be connected, but the semantic of those elements is defined in the meta model of each method.

That's why it is important to map the meta model elements of both exchange partners to have a clear definition how SysML is used at both exchange partners.

Before starting the exchange process, it is important to baseline the architecture model. Only with a baseline it is clearly defined what the export includes. Especially for a round trip exchange scenario, where the receiver of an architecture model refines the content and sends back the content to the owner, it becomes nearly impossible to merge the content into the original model. Mostly the content at the provider side has also changed, improved and modified. That's why a baseline concept is mandatory to be able to compare both models and merge the elements from the OEM with the modified elements of the supplier. The LieberLieber demonstrator shows how such a baselining and merging approach between an OEM and supplier model can look like.

Next to the methodical alignment it is a challenge to define the parts of the model that should be exchanged. In case of a flat model structure, it is very challenging to cut out the elements for an export, because in each architecture model elements are highly interconnected. Splitting the model into packages improves the capability to exchange only parts of a model. A package structure helps the architect to structure the model into parts which can be easily exported and parts which should remain internal. However, also packages will have relations to other packages, but it's a good approach to reduce dependencies between model elements and define the brake point where a model relation will be broken for the model export. With packages it is clearly defined what is included in the export and especially what is not part of the export.

A useful approach to enable the exchange process is to distinguish between specification and design elements. A specification element can be defined by the OEM and should include all required black-box information like black-box functions, interfaces, state machines etc. which are required by the supplier to refine the sub-system. The specification can be managed in one package and exported for the supplier. Figure 57 shows how specification and design elements can be separated by packages.

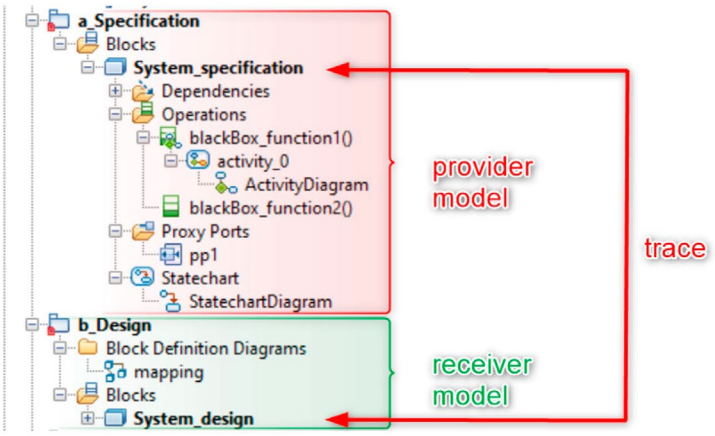


Figure 57: Separation of specification and design element in different packages

A specification should always be imported as read only to ensure that the specification from the OEM will not be changed.

The supplier is responsible for the design element and can use the specification from the OEM to specialize its design element (see left Figure 58 below). In terms of using different methods and SysML elements a mapping view helps to create a trace between the elements of both models. Therefore, the SysML dependency relation can be used (see right Figure 58 below). With this approach a clear separation between the OEM and supplier model elements is defined and the SysML generalization can be used to reuse pre-defined structure elements including its behavior and interfaces in the model of the supplier.

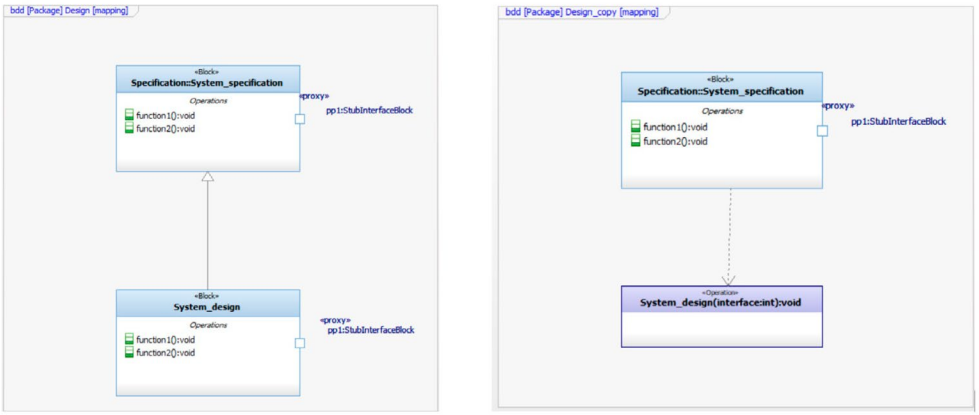


Figure 58: Specialization of the specification element (left) and a trace between specification and design element in case of different methods (right)

To ensure that the exported content can be read and imported at the receiver side only the aligned SysML elements and diagrams should be used. The required minimal set of elements is defined in Figure 14. A known issue of the model exchange is that diagrams and geometrical coordinates of model elements can't be exported. In consequence diagrams can't be transferred to the receiver of the model. However, next to the model content the author of an architecture always provides additional information in the way how a diagram looks like and what elements are shown at a diagram. This means also the visual representation of a model is a very important input for the receiver of an architecture. As long as the exchange of diagrams is not possible, a good practice is to provide next to the model data also pictures of the required diagrams for a better understanding.

A mechanism to customize the engineering process is to extend and modify the SysML language by project specific profiles. the workflow forum has identified that each company has their own profiles. For the exchange process it must be decided if the profile will be exported with the model. If the model is exported including the profile, all required

extensions will be available at the receiver side. Before the import the receiver of the model needs to ensure that the extensions coming with the exported profile will not contradict the settings at the receiver side. If the model will be exported without the profile, all attributes and extensions defined by the profile will be lost and can't be used on the receiver side, but conflicts with the settings at the receiver model can be avoided.

SysML is also a profile based on UML. It is recommended to not modify the SysML profile, because if the provider exports its model including the project specific modifications of the SysML profile, also the pure SysML settings will be overwritten at the import side. Especially if both parties work with different SysML versions. This means all company specific extensions should be managed in a separate profile next to the SysML profile.

Most SysML authoring tools provide also capabilities to adapt the tool to the needs of the engineering domains. To improve engineering efficiency this feature is very often used by the user companies. Therefore, project specific settings are often part of the model, like perspectives or profiles.

Sometimes tool customizations and SysML extensions are stored in one profile. If the OEM and the supplier uses same tools, but different tool customizations, the exported profile can contradict the setting of the supplier. That's why it is highly recommended to split tool customization and profile into different files.

In addition to the exchange of architecture models it is essential to exchange requirements between the OEM and supplier. Today requirements are usually managed by RQM tools next to the architecture models. Requirements are mostly documented in text format and are linked to SysML elements in the architecture models, e.g. by OSLC technology. However, this means requirements and architecture are managed in different tools which makes the exchange process complex, especially if also links should be exchanged. A reference exchange process of requirements between the OEM and the supplier is defined by the ReqIF workflow forum. This process considers only the exchange of requirements, SysML elements and links between both are not in scope. In consequence links will be lost during the model transfer. Today there is no process defined to exchange requirements, architecture models and links between both. Tools like shown in the MID demonstrator can support the user to trace requirements and architecture in an additional web view, but do not support the exchange itself. In order to improve the model-based collaboration, specifications of new data formats like SpecIF, OpenMBEE and MTIP are developed, but today's SysML tools do not support them for a productive usage.

However, since there are no standardized solutions available at the market, the SysML WF has defined a set of requirements towards the exchange process. The high-level requirements are documented in chapter 3.2. A recommendation for the exchange itself is not available.

In future there will be an additional trend where more and more requirements are managed in the SysML tools. In consequence SysML tools should also support the exchange of those elements and provide capabilities to manage requirements. In this case also links become model elements and must be exchanged as well.

7.1.2 Recommendations to Tool Vendors

Instead of opening collaborative modelling tools across company borders, we would like to hand over models in a tool independent and file-based manner as it meanwhile works for requirements based on ReqIF. We would recommend to the tool vendors to agree on an efficient, well documented, and publicly accessible file format.

At the moment five file formats seem to be technically capable of transporting SysML models,

- the XML Metadata Interchange (XMI) standard, published by OMG,
- the data format used by OpenMBEE between their MDKs and their MMS,
- the Open MBSE Data Package, implemented by MID GmbH,
- the Specification Integration Facility (SpecIF), supported by GfSE e.V.,
- the Modeling Tool Integration Plugins (MTIP), implemented by The Aerospace Corporation.

We recommend that the selected format is also capable to refer to other structured engineering data, stored outside the SysML model, like related requirements, parameters, or test cases.

An open-source format seems to be most promising to guarantee transparency and to adapt to future demands e.g., upcoming SysML versions.

Besides the question for a suitable file format, we recommend that it is possible to exchange only specific parts of a SysML model. This becomes necessary for protection of intellectual property, but also to be able to work on different parts of a model with different partners in parallel. The exporting person has to be in the position to keep track which parts of a model are going to be exported.

In the course of an import of a SysML model, an elaborated diff & merge mechanism or at least a preview of the changes is necessary to avoid importing wrong or corrupted data.

7.1.3 Recommendations to Committees (VDA/ivip)

SysML-based collaboration processes with today's existing exchange formats cannot be performed with sufficient quality to ensure digital continuity. The example of XMI-based exchange shows that the capabilities are limited (see Chapter. 5.1.1).

Collaboration processes conducted with proprietary data would avoid information loss, but cannot be combined with today's heterogeneous system landscapes. Furthermore, readability of proprietary formats in downstream processes cannot be assured. This applies to both internal and external cross-company collaboration.

To address these challenges, the users of the SysML-WF have concluded that flexible system environments require the use of neutral exchange formats.

In order to define a neutral format for the exchange of SysML system models, the working groups can basically imagine two ways, which can be pursued in committees as for example the prostep ivip and the VDA:

- the development of a new exchange format specifically designed for the use cases defined in the SysML WF or
- the further development of an already existing format (see Chapter. 5.1) such as e.g. SpecIF or ReqIF (e.g. extension to link requirements to external artefacts).

7.2 Next Steps

The next steps to continue the **SysML WF** are being discussed and are not yet finally confirmed, but a continuation in the current form is not planned. Nevertheless, one approach for continuing the WF activities could be to have them executed by other prostep ivip projects in the future, as there are many opportunities to benefit from synergies.

One possible way could be to consolidate the contributions of several overlapping WFs, including the SysML WF. Currently, the possibility of establishing a common prostep ivip IF on the topic of digital twin is being discussed. The WF topics including SysML WF could be addressed in several separate work packages within this common project. One benefit of such a consolidated project is, that these work packages could generate input for one common IF demonstrator in order to benefit from synergies. It is planned to establish a close collaboration between this consolidated ivip project and CATENA-X.

From **SysML IF** perspective the continuation in the current way is also not yet decided. Most topics demand a close collaboration with the SysML WF and their members. Therefore, it is essential to plan the continuation in close coordination with the SysML WF.

The idea of a "common" IF mentioned before, not only focusing on system model exchange, is still under discussion. Discussion will be continued during one of the next joint workshops. A common IF would bring the benefit, that vendors can enhance their demonstrators in a way, addressing the needs of more than one working group. On the other hand, it might also lead to a loss of the specific technical focus.

One possible next step could be the establishment of smaller working teams between SysML WF and IF members for working on the demonstrators. This idea has already been discussed and considered as a good approach, but could not be established so far due to different reasons.

Another next step from the IF could be to run a survey to get feedback on the potential formats and further strategies on model exchange from vendor perspective. This input could then be used to plan a future benchmark to evaluate the current capabilities and potentials of the different exchange formats. Focusing on one of the formats, based on the benchmark results, could guide the demonstrators into a common direction.

8 Summary and Outlook

The goal of this paper is to point out the current and future role of tool interoperability based on SysML in MBSE with a major focus on SysML v 1.6 as standard format.

Starting with an overview of the general aspects of the topics of Systems Engineering and SysML, this recommendation shows ways of dealing with the defined use cases. Besides the use cases "Requirements Engineering & System Design" and "Verification & Validation", the main focus is on the use case "Model Exchange". Within this use case, the desirable workflow and content of the Model Exchange is described, and the topic of IP protection is considered.

Based on the requirements of the SysML WF, demonstrators have been developed in the SysML-IF, which were first described and subsequently evaluated. In addition, already existing exchange formats such as XMI, SpecIF and Project MTIP are presented.

With regard to the paper's goal of highlighting the current and future role of tool interoperability, a broad overview of current, promising approaches was compiled. In the end there is currently no fully developed format in the sense of broad interoperability and no clearly defined collaboration workflow. Therefore, based on the concepts presented, this paper concludes with recommendations to the collaboration workflow, tool vendors and committees regarding future development.

In order to drive future development accordingly, an outlook on the topics to be dealt with in the future would be:

- Expansion of activities referring to the Model Exchange use case like:
  - Managing IP protection during model exchange
  - Handling of subsets
  - Roundtrip model exchange capabilities
- Continue with use case "Requirements Engineering & System Design"
- Continue with use case "Verification & Validation"
- Further investigation of SysML v2 regarding Model Exchange
- Work on an automotive-specific profile for SysML
- Benchmark of existing SysML exchange formats

The working group sessions showed, that the exchange of system models is a lot more complex, than it might appear on first sight. The high complexity and traceability inside the system models in combination with company specific modeling methods and the different implementations of SysML within authoring tools, mean a big challenge for defining a possible future standard. The further use cases describing the exchange of system models in combination with linked requirements and V&V artefacts will bring in additional complexity. To find a common solution for model exchange in the future, it demands effort and a clear commitment of all involved parties.

9 Bibliography

Dumitrescu, R., Albers, A., Riedel, O., Stark, R., & Gausemeier, J. (. (2021). *Engineering in Deutschland – Status quo in Wirtschaft und Wissenschaft*. Paderborn.

Dungern, O. v. (2016). Semantic Model Integration for System Specification. *TdSE Tag des Systes Engineering der GfSE*. Herzogenaurach.

Dungern, O. v. (April 2022). Kollaboration im Systems Engineering mit ReqIF. *REConf*. München.

Dungern, O. v. (November 2015). Integration von Systemmodellen mit fünf fundamentalen Elementtypen. *TdSE Tag des Systems Engineering der GfSE*. Ulm.

Dungern, O. v. (November 2020). Specification Integration Facility – Wozu braucht man SpecIF neben SysML. *TdSE Tag des Systems Engineering der GfSE*. online.

Friedenthal, S., Moore, A., & Steiner, R. (2011). *A Practical Guide to SysML, 2nd Edition*.

INCOSE: Systems Engineering Definition. (2019). Von https://www.incose.org/about-systems-engineering/system-and-se-definition/systems-engineering-definition abgerufen

Kaufmann, U. P. (2015). 10 Theses About MBSE and PLM. *Gesellschaft für Systems Engineering*.

Kleiner, S. H., Lindemann, G., Korobov, S., & Hamester, M. (2018). Use Case driven Model-based Systems Engineering for industrial applications. *Tag des Systems Engineering 2018*.

Object Management Group. (2019). *SysML 1.6*.

OMG SysML. (2022). Von https://www.omgsysml.org/what-is-sysml.htm abgerufen

OpenMBEE. (01 2022). Von https://www.openmbee.org/index.html abgerufen

Severson, T. M. (2022). *Modeling Tool Integration Plugins (MTIP)*. The Aerospace Cooperation.

Specification Integration Facility (SpecIF). (2022). Von https://specif.de/en/ abgerufen

The Aerospace Corporation – Github. (2022). Von https://github.com/the-aerospace-corporation abgerufen

The Aerospace Corporation. (2022). About us. Von https://aerospace.org/about abgerufen

VDI/VDE 2206:2021-11. (2021). *Development of mechatronic and cyber-physical systems*.

Weilkiens, T. (2019). SysML v2 – The Next Generation. *ReConf 2019*. München.

prostep IVIP

prostep ivip Association

Dolivostraße 11
64293 Darmstadt
Germany

Phone +49-6151-9287336
Fax +49-6151-9287326

psev@prostep.com
www.prostep.org

ISBN 978-3-948988-23-4
PSI 28
2023-02/Version 1.0