 THE RESEARCH LEADING TO THESE RESULTS HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S SEVENTH FRAMEWORK PROGRAM (FP7/2007-2013) FOR CRYSTAL – CRITICAL SYSTEM ENGINEERING ACCELERATION JOINT UNDERTAKING UNDER GRANT AGREEMENT N° 332830 AND FROM SPECIFIC NATIONAL PROGRAMS AND / OR FUNDING AUTHORITIES.

CRitical S**YST**em Engineering **A**cce**L**eration

WP Report:
Functional Powertrain Architecture & Control Development with respect to Integrated System, Safety and Requirements Engineering

D303.011

DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Title	WP Report: Functional Powertrain Architecture & Control Development with respect to Integrated System, Safety and Requirements Engineering
Deliverable No.	D303.011
Document Version	V1.0
Date	2014-01-31
Contact	Dr. Michael Maletz
Organization	AVL
Phone	+43 316 787 2139
E-Mail	michael.maletz@avl.com

AUTHORS TABLE

Name	Company	E-Mail
Michael Maletz	AVL	michael.maletz@avl.com
Alexander Stiegler	AVL	alexander.stiegler@avl.com
Julia Weber	AVL	julia.weber@avl.com
Thomas Rüter	AVL-S	thomas.rueter@avl.com
Martin Baum	AVL-S	martin.baun@avl.com
Selver Softic	ViF	selver.softic@v2c2.at
Nadja Marko	ViF	nadja.marko@v2c2.at
Egbert Althammer	AIT	egbert.althammer@ait.ac.at
Holger Kampffmeyer	PTC	hkampffmeyer@ptc.com
Sören Kemann	FhG-IESE	soeren.kemmann@iese.fraunhofer.de
Serrie Chapman	IFX-UK	Serrie.chapman-EE@infineon.com
Christian Hein	FhG-FOKUS	christian.hein@fokus.fraunhofer.de
Grit Dudeck	IST	dudeck@testingtech.com

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
0.01	17.10.2013	New document	
0.02	08.01.2013	Input from WP partners included in document – start of internal Review	
0.02	20.01.2013	Internal WP review comments included; additional input from UC partners includes – start of external WP review	
1.0	31.01.2014	External review comments incorporated	

CONTENT

List of Figures

1 Project objectives for period M1- M9

This chapter describes the objective of the work package from the CRYSTAL project point of view (in terms of IOS & RTP), as well as from the use case content point of view (industrialization of processes, methods & tools).

1.1 General Work Package Objectives

The main objective of the work package for the period M1 to M9 is to detail the use case. This is done by a clear and detailed definition of the use case framework as applied within AVL, as well as taking aspects concerning relevant engineering methods and bricks from the use case partners into consideration.

As requested by SP6, all use cases and according descriptions shall follow the same structure. This is then the basis for the interoperability needs capturing and shall serve as an input for the IOS working group. Figure 1 outlines the specific steps given by IOS group which are also applied for the use case

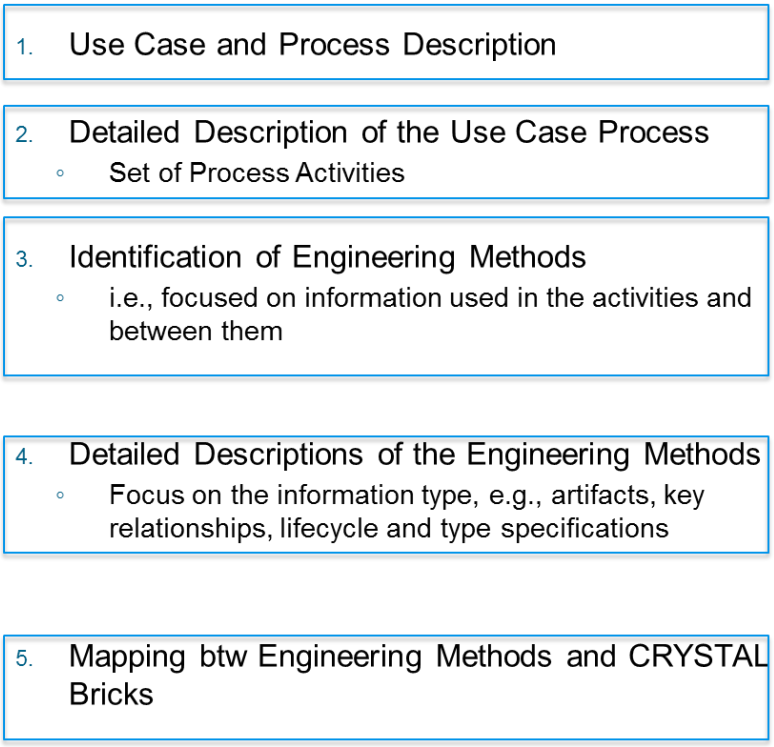

Figure 1: WP Steps

The focus of this first period is on steps 1 to 3. They include the use case and process description, a detailed description of the process activities, as well as the identification of the main engineering methods.

Further on, the objective is to detail selected engineering methods based on a provided template to capture interoperability requirements which was conducted within this work package.

1.2 Objectives in terms of Use Case Content Industrialization

This use case focuses on integrated system, safety & requirements engineering within automotive powertrain and control development. It is based on tools which are currently applied within AVL, as well as on tools that are considered for future development environments.

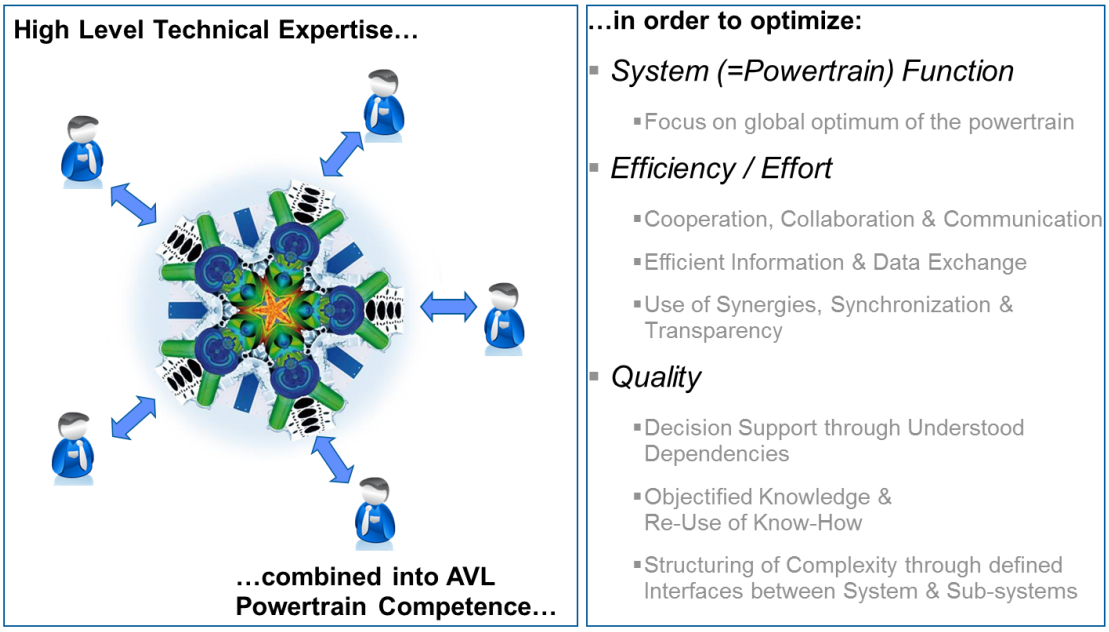

Figure 1-2: Overall Goal of Use Case

The objective of this use case is to increase the quality and efficiency of powertrain systems development and safety activities by applying model-based systems engineering (MBSE). These activities cover the development process with respect to requirements engineering and systems design/analysis, the support of discipline specific development activities, as well as the integration and validation of the system under development. In order to reach this objective, existing (and emerging) methods and tools to support MBSE must be brought to a maturity level that can be applied/rolled-out in an industrial environment and integrated into tool chains in order to support users in terms of engineering and support activities.

2 Work *Package* progress and achievements during period M1-M9

As mention in the previous chapter, the objective is to detail the use case content in order to derive interoperability requirements. The following table summarizes the progress and achievement made.

Objective	Progress & Achievements
Use Case and Process Description	Use Case detailed based on content as defined in project proposal. According description of the development process within AVL fully described. Tools with respect to interoperability within AVL systems engineering identified.
Detailed Description of the Use Case Process incl. Set of Process Activities	Applied process activities identified and described. According main deliverables of process activities identified and detailed.
Identification of Engineering Methods, i.e. focused on information used in the activities and between them	Engineering methods required for efficient execution of process activities identified. Engineering methods described in terms of purpose, input/output, supported tools, as well as supporting sub-activities/methods.
Detailed Descriptions of the Engineering Methods; Focus on the information type, e.g., artifacts, key relationships, lifecycle and type specifications	Selected engineering methods and according tools described in detail. First interoperability needs / requirements derived.

Based on the above mentioned objectives and achievements, first interoperability scenarios and requirements were defined. These shall be communicated to SP6 and serve as input for detailed SP6 activities as defined.

3 Use Case Framework

Powertrain development has become more complex not only because of the technological challenges along hybridization and electrification. The process of developing and optimizing a complete powertrain requires skills from different engineering disciplines and, therefore, requires a joint and transparent development process, as well as a tool environment which efficiently supports the execution of these processes.

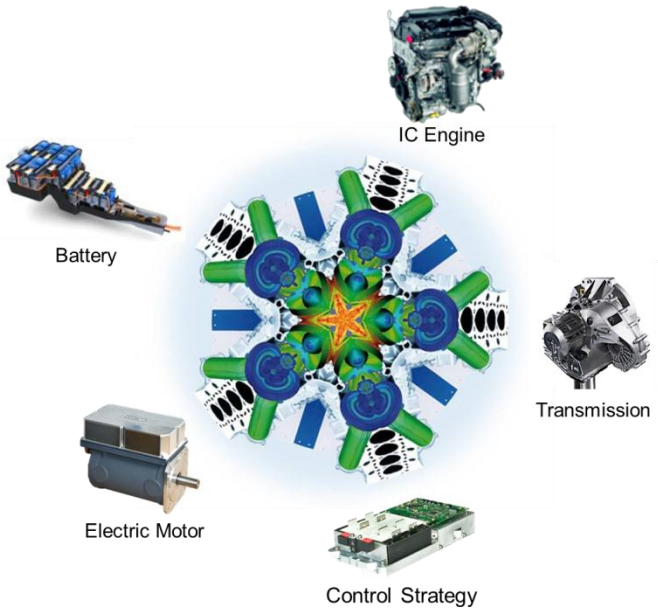

Figure 3-1: Powertrain Elements of Use Case

The powertrain development process described in this use case represents a coordinating process which links the project management process and the respective engine, transmission, battery, e-machine, and control system processes. It also includes the definition of operative systems engineering activities (i.e. system design & requirements engineering, system simulation, system integration, system safety, verification & validation). Powertrain system development activities to achieve the global powertrain optimum should synchronize downstream activities of the powertrain element development processes with upstream activities of the project management process without duplicating activities of those areas.

The AVL powertrain development process combines experiences from different skill areas and processes. The hardware driven development of product generations allows increasing maturity over time. The development phases within each generation are closely linked to the V-Cycle, which is typically used in software and system development. The combination of development process and V-Cycle allows the joint development of design, software, and system engineers which is the focus of this use case.

3.1 Use Case Description

Step 1 of the use case is a technical order from the customer to develop a) a new powertrain solution or b) perform system, safety & requirements engineering activities driven by specific vehicle goals and performance criteria based on existing solutions. This task includes technical and organizational documentation, which is available through tool supported PLM solution generally provided in office formats (i.e. word, excel, ppt, pdf etc.). This includes the collection of

initial stakeholder objectives and boundary conditions concerning the targets of the system under development.

According tool support in this step is provided through the requirements engineering tool PTC Integrity. Furthermore, respective safety information to fulfill ISO26262 (e.g. for the following hazard and risk assessment, elicitation of safety goals, etc.) is assessed during this requirements analysis phase.

Step 2 of the use case focuses on the modeling and analysis of the target system for a detailed investigation of customer needs, dependencies, and interactions. The creation of a model-based user requirements specification, including a preliminary architecture definition with SysML, is the base step for all subsequent requirements engineering activities.

The verification of the architecture and according operation / control strategy is supported by vehicle and powertrain simulations (i.e. using tools such as InMotion & AVL Cruise) which also give input to the functional safety concept. The findings result in a detailed investigation and further development of the initially defined architecture incl. requirements cascading and breakdown - creation of model-based system specification (also with SysML tools). Technical system design details such as the allocation of interdependencies between requirements, functions, and structure with the powertrain / control architecture, HW/SW interfaces or the technical safety concept with additional emerging technical safety requirements are in the focus of this use case step.

Step 3 focuses on the collaboration and communication of requirements to discipline specific project teams – locally distributed over several countries. PLM tools (e.g. PTC Windchill and Integrity, SharePoint, etc.) support this activity including the investigation, further breakdown, and detailing of these specific requirements with a tight integration of a model-based system development approach (sub-system / component specifications) embedded within the PLM environment. Safety related activities as for example system and safety analysis, FMEA, FTA, etc. are supported by adequate and integrated tools (e.g. APIS IQ-FMEA, ReliaSoft Blocksim, SysML, EAST-ADL, etc.) and safety methods. In the following detailed system and safety design regarding e.g. HW/SW interfaces or technical safety concept is performed.

Discipline- and team-specific solutions in terms of models, simulations, and analysis are usually supported by discipline-specific authoring tools (e.g. Matlab), as well as a common tool chain for collaboration (e.g. exchange of model parameters, co-simulation, document/ data, and process management). The discipline specific analysis also provides additional input to e.g. the specification of SW & HW safety requirements, architecture design, HW & SW design etc.. An appropriate level of integration for these development activities and involved tools is required in order to investigate dependencies between discipline specific solution models with respect to the multidisciplinary target system (e.g. feedback of input/output parameter values).

Solving identified technical conflicts and discrepancies including the assignment and tracking of the solution progress, the verification of safety requirements and system design, and an adequate decision support finalizes this use case. This is achieved by continuous and adequate tool support for documentation and collaboration throughout all use case steps.

4 Detailed Description of Use Case

This chapter describes the details of the use case. This includes the description of the use case content, a process description and its integrated development activities. Based on these activities, the main deliverables from systems engineering point of view are identified and described. These are then the source for the main development and engineering methods that were identified and applied within systems engineering.

4.1 System Development Description & Context

4.1.1 Goals & Challenges of Systems Engineering Approach

The development of a product under the premises of high quality, efficiency, and functionality is the overall goal of systems engineering. To achieve this goal, the whole powertrain competence has to be integrated into a common process – the powertrain development process. This process aims to optimize system functions to get the global optimum, to optimize the system efficiency, and the quality.

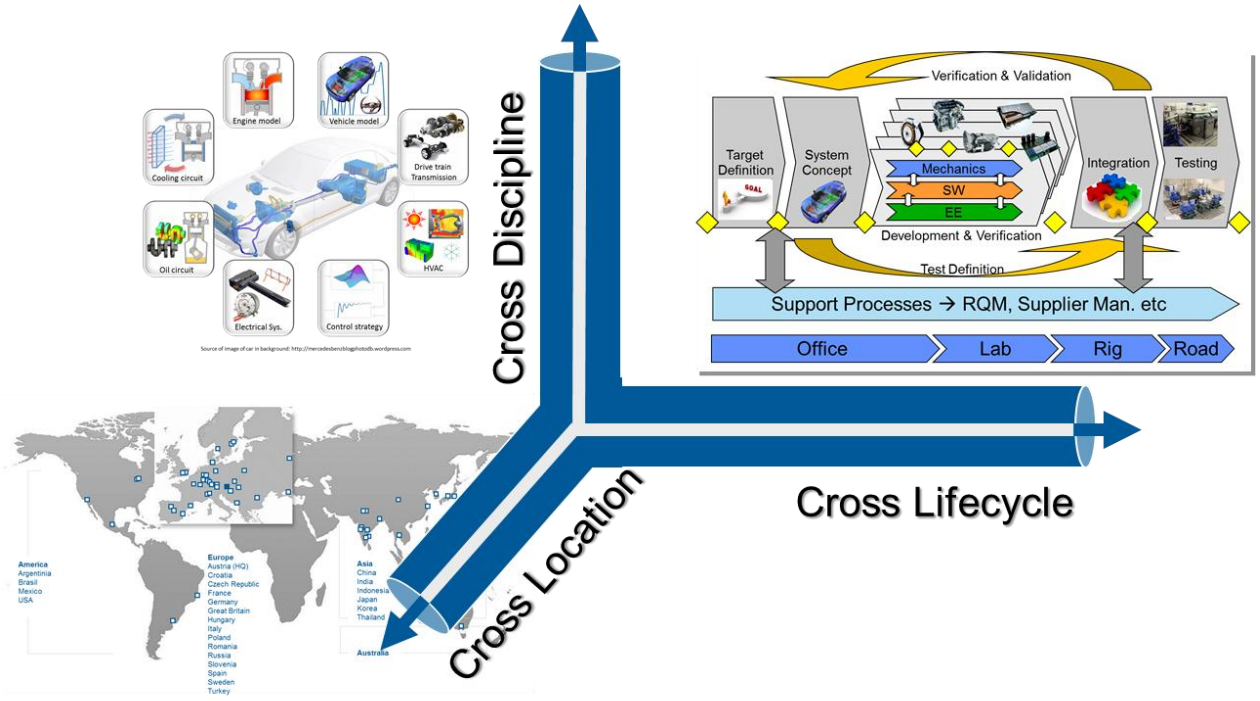

Figure 4-1: Challenges of the powertrain engineering process

The increasing complexity, divergent vehicle targets, and a heterogeneous development environment are the motivation for an integrated powertrain development process. The increasing complexity is shown in the rising number of vehicle variants, topologies and architectures, and new vehicle functionalities. As shown in Figure 4-1, the different selling markets, the product lifecycle, and the need for a seamless cooperation between different disciplines and their methods and tools are the major challenges.

The view on systems engineering as a discipline can be a solution to cope with these goals challenges. As shown in Figure 4-2, systems engineering is the central discipline to coordinate the

vehicle / powertrain development activities to achieve the vehicle targets. The approach is to follow the systems engineering activities in order to develop requirements for the subsequent development activities of sub-teams. These sub-teams are responsible for the development of components in terms of hardware and software development (e.g. engine incl. design of assemblies, control system such as HCU incl. control HW and SW system, etc.). Further on, systems engineering activities define acceptance criteria for requirements (e.g. the implementation of certain parameter thresholds during development) , as well as test case (on system level) based on use cases. Therefore systems engineering closes the gap between hardware and software related development activities in early lifecycle phases and supports related integration and calibration activities.

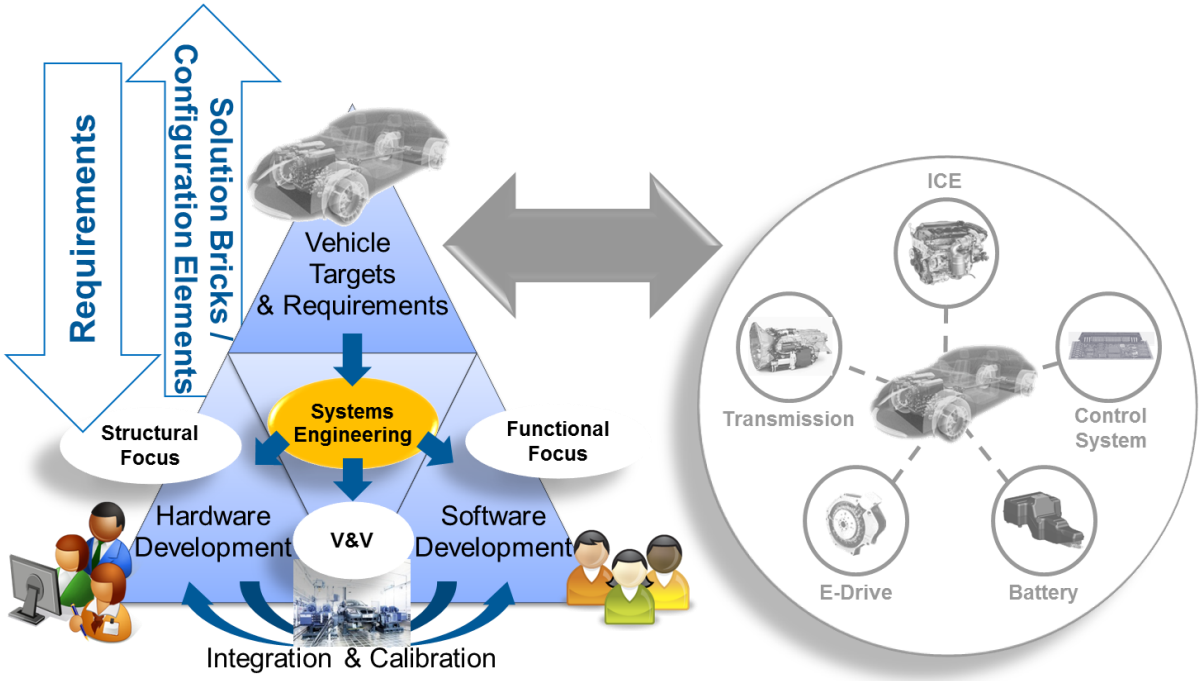

Figure 4-2: Systems Engineering as a Discipline

The following goals are in the scope of these systems engineering activities:

- Delivery of technological innovation on powertrain system level
- Combination of hardware and software/functional design – alignment of activities via phases and generations
- Systems engineering coordination in addition to operational activities
- Demonstrate structure and quality of services delivered by AVL engineering
- Continue to enhance system methodology (Requirements, Simulation, and Safety) to support powertrain development
- Enabler for frontloading
- Drive efficiency and frontloading into AVL development processes
- Modularity of product development process (focus on engineering service provider specific needs)

4.1.2 System Development Description

The result of the related development process is a system design for hybrid/electric powertrains, which are used e.g. for passenger cars, commercial, or light/medium/heavy duty vehicles.

Overall interoperability challenges are:

- all involved development tools should share a common schema (e.g. input-output interfaces)
- seamless integration of activities & tools, incl. safety within heterogeneous development environments
- continuous data & information exchange between system design & testing/test bench

Figure 4-3 shows the V-model for powertrain system development. Several concepts for powertrain system design are developed based on the definition of targets for the powertrain system (based on vehicle goals). The concept development activities include the development of initial requirements and use cases. The system design activities include the definition of architecture incl. powertrain topology, system features (resp. hybrid control features), and the specification of main components. This information is then used as an input for component development activities. Systems engineering supports these component development activities with coordinating activities, i.e. the consolidation and exchange of system relevant data and information. Systems engineering also plays an important role for integration aspects in terms of system verification and validation, especially functional testing of powertrain systems.

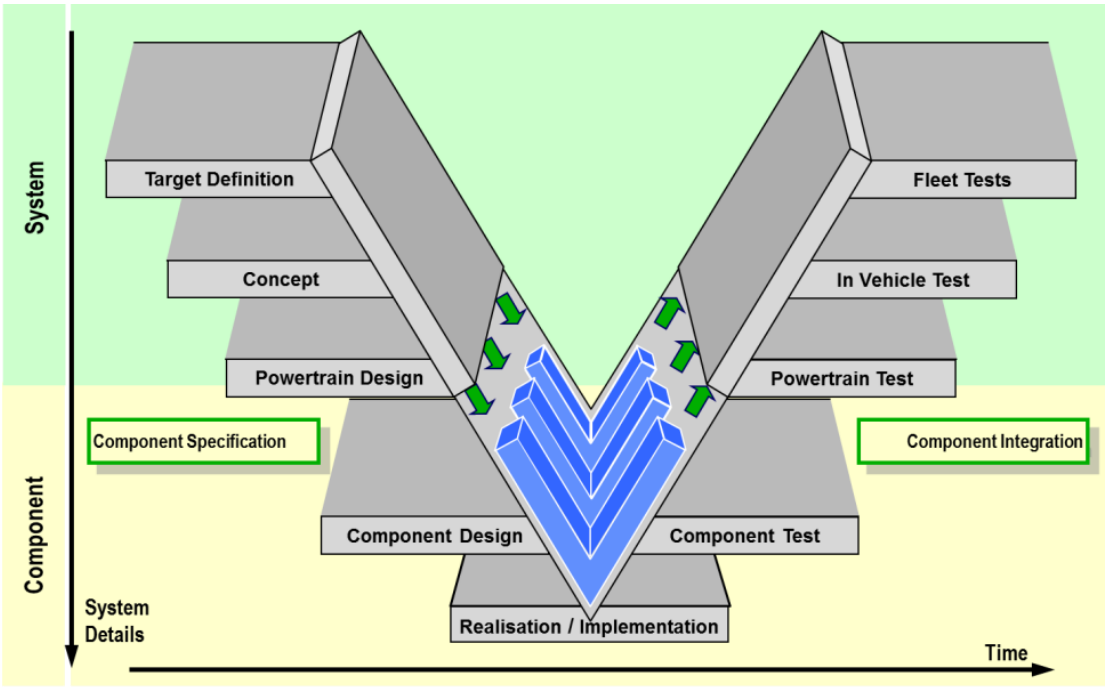

Figure 4-3: V-Model view of Powertrain System Development

With a slightly different view onto the V-Model, as shown in Figure 4-4, there are different disciplines to pass through during the development. Industrialized model-based systems engineering for integrated system, safety, and requirements engineering within powertrain & control development helps to describe the system for all cross-disciplines, especially the hardware

components (e.g. mechanics, wiring etc.) and software development but also for system validation & verification and system integration.

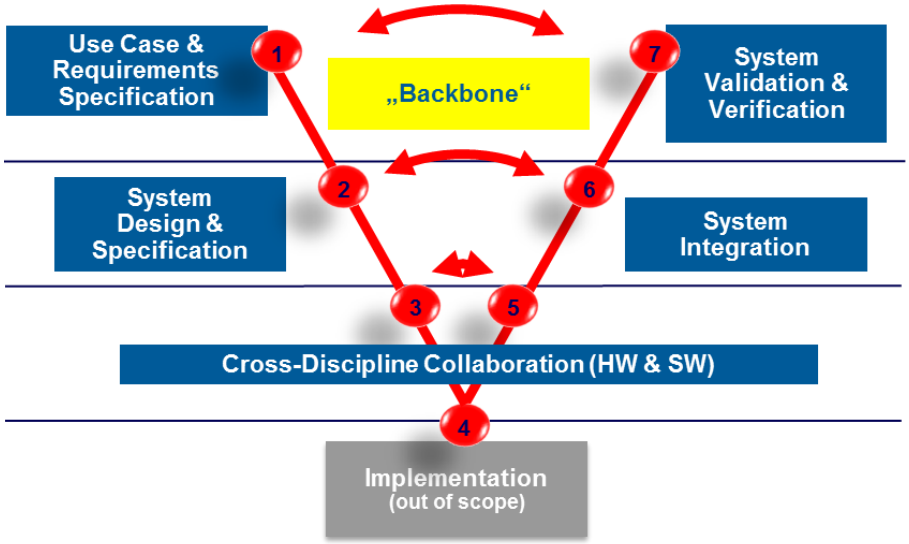

Figure 4-4: Systems Engineering in the V-Model

4.1.3 System Development Framework

The framework for system development is based on requirements engineering and a function-oriented and integrated system model. Requirements engineering takes use cases, vehicle targets, performance criteria, standards, etc. as input and generates a holistic, formal system description. The description includes the refinement of vehicle requirements (RQ's) down to component RQs.

The RQs are used into the system model for detailing. This system model includes components, parameters, interfaces, features/functions, system behavior, operation modes, transitions, and other dependencies defined by internal model-links to describe the system. The system model can be used to generate different system views, which include the information and interface description required by the single engineering disciplines on the subsequent level.

The use case framework is shown in Figure 4-5.

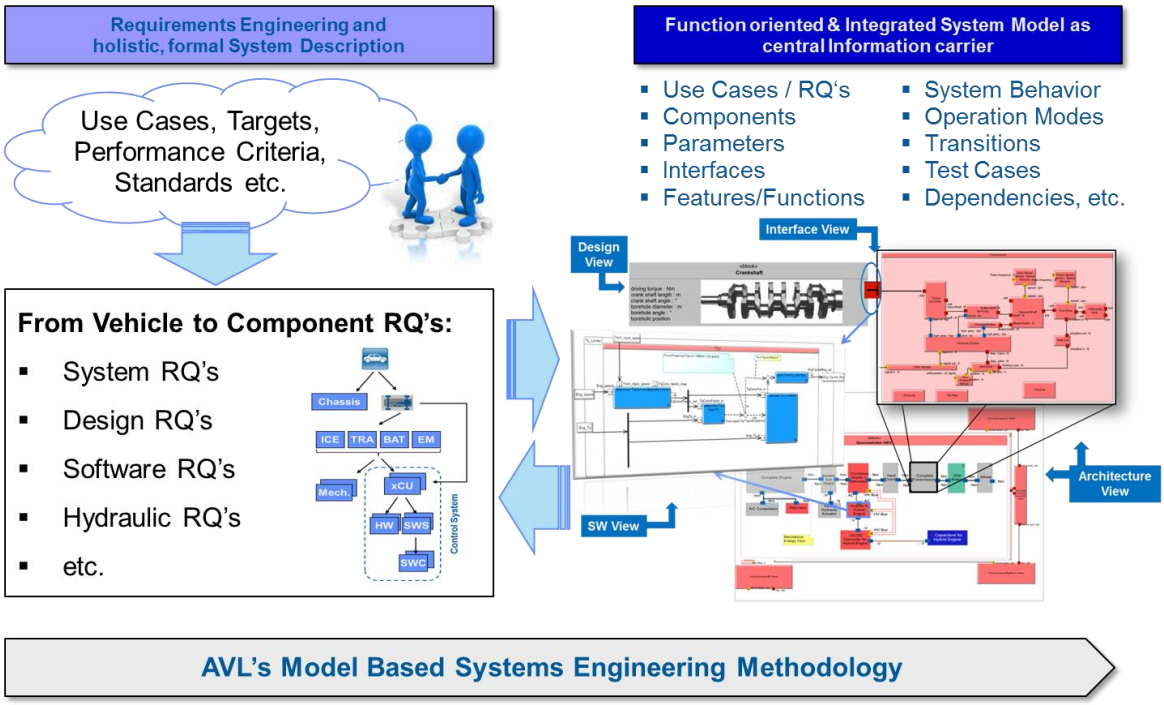

Figure 4-5: Use Case Scope & Framework for MBSE

4.1.4 Objectives for Industrialization

There are several aspects which have to be considered for a productive and industrialized application of such systems engineering methods and tools. These are:

- Define, specify, implement, and verify tool chain (incl. required methods, etc.) for a defined use case. This includes:
  - Definition of required methodology
  - Specification of requirements for tool implementation based on methodology
  - Development of respective meta-models
  - Definition of interoperability aspects incl. definition of data and information exchange processes
  - Definition of implementation test cases and acceptance criteria
  - According documentation of implementation
- Support efficient development, documentation, and life-cycle management of systems engineering deliverables
- Correlations & synergies between systems and safety deliverables
- Define method & tool interoperability for deliverables (content in which tool, in which level of detail, dependencies etc.)
- Show consistency between
  - System definition & specification (e.g. UC, RQ's, Function description)
  - System specification & simulation (e.g. exchange of parameters for simulation)
  - Development / realization of solutions (e.g. HW components, control functions)
  - System Integration, V&V (e.g. TC, test planning & management)

- Prove benefits of applying integrated tool-chain & backbone

4.2 Development Activities

The realization of the UC framework described above includes several development activities. These development activities focus on systems engineering and serve as an input for subsequent development activities within individual departments. Systems engineering activities themselves are also distributed all over the organization and therefore require a close collaboration of the involved development teams.

Figure 4-6 outlines the main development activities taken into consideration for this use case. It serves as an overview in order to understand how these activities are related, however, it is not the intention to detail every activity. The intention is to focus on activities most relevant for systems engineering or MBSE, respectively.

The following chapters give an overview of the individual activities of Figure 4-6.

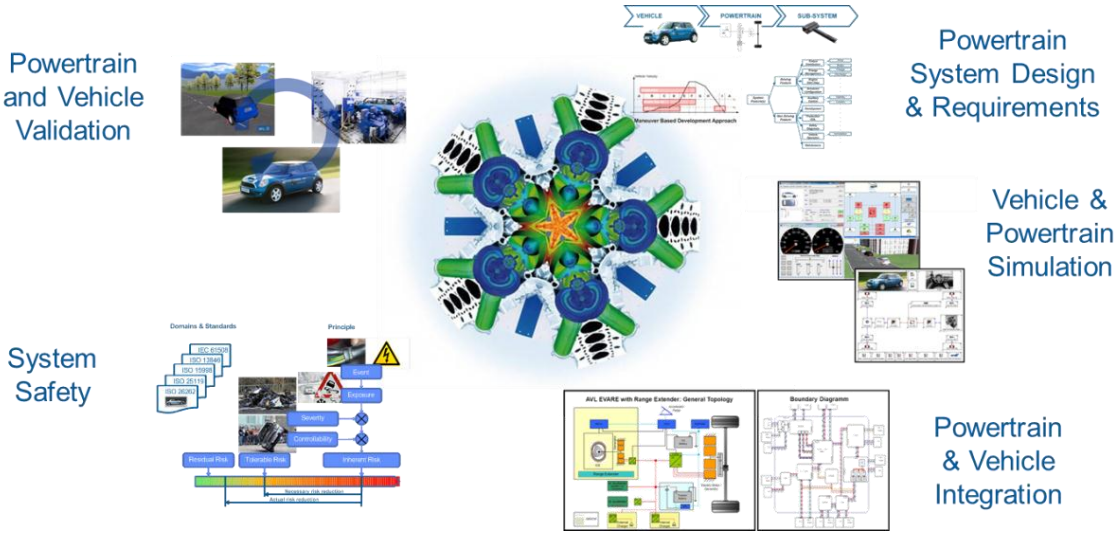

Figure 4-6: Overview Development Activities of UC

4.2.1 Powertrain System Design & Requirements Engineering

The powertrain system design and requirements engineering includes several activities which have to proceed throughout the development process. The first action is the cascading of the vehicle level requirements to powertrain system and element level requirements.

For detailing of the vehicle requirements customer Use Cases (UC) are defined which consider various boundary conditions for the requirements and serve as discussion base with the customer A resulting UC describes the “Load Conditions” (e.g. ON/OFF durations) for the vehicle, powertrain, and components due to system operation (driving) and non-operation (shipping, maintenance, rescue, production) over the full lifecycle (development, production = End of Line, user operation, maintenance, service & component replacement, End of Life) with respect to the influencing factors coming from the environment, the operation (= driver, driving profile). The UC may also include the variation out of the components (e.g. altering or temperature dependencies). After the customer confirmed that all development participants have the same understanding of the vehicle requirements, the system feature development can start.

The System Features (e.g. Engine Start/Stop or pure electric driving) are developed systematically based on the targets (requirements) for the vehicle & the expected system attributes. The features

are described on system top level, in system context in a maneuver-based approach (e.g. Start/Stop sequence in city driving). Initially this approach is not SW & SW Architecture related. In a next step the system requirements are derived from the system feature description (e.g. which energy capacities are required for pure electric driving for the specified target distance)

The System RQs are the base for further element specifications (hardware as well as software). System RQs are derived as consolidated RQs from different sources (feature development, system development / calibration, functional safety, system quality, reliability).

4.2.2 Vehicle & Powertrain Simulation

The vehicle and powertrain simulation consists of different activity steps. Simulation activities are performed throughout the entire vehicle and powertrain development process, starting from a first simulation of a basic vehicle, over to a first hybrid vehicle simulation to several updates of the previous simulations. The simulation results are used to check the targets, optimize the system, and for verification & validation.

As one key aspect, the system simulation provides generates the fuel consumption value or the fuel reduction compared to the base non- hybrid/EV vehicle, respectively by introducing hybrid components and the appropriate hybrid strategy. Furthermore, other properties, such as the influence of component size and efficiency, non-driving components (auxiliaries), functionality, and calibration could be assessed. The output of this task is documents and data, which describe and define the powertrain system. Additionally, an initial validation of the vehicle targets is achieved. As part of the definition process, content has been clarified with the customer and has to be used in downstream activities to specify and develop the powertrain.

4.2.3 Powertrain & Vehicle Integration

Powertrain system integration activities include the coordination of the different hardware component and software integration tasks into the vehicle. The Functional Implementation Plan (FIP) summarizes the coordination planning and includes all input data required for the powertrain integration.

The development and execution of the FIP is a mandatory step in order to ensure an aligned development and failure free implementation of all powertrain related functions. It defines a common timing which all parties involved in the development, implementation and verification will be able to rely on and schedule / plan development resources efficiently. It also includes information about the features / functional maturity of the powertrain system over time for the communication of the development progress to the customer.

The FIP will be updated along to the development steps (A-, B-, C- Sample) up to SOP.

4.2.4 System Safety

Each sub-system of the top level system (e.g. a cooling circuit) that is able to cause failures which could harm people is called safety relevant system. This classification is usually done by the customer. The customer provides the already identified safety-critical architectural elements, or at least the already depicted safety goals. A safety goal is a top level functional safety requirement (e.g. “avoid unintended torque”). The system safety developer/functional safety manager should start a quantitative safety analysis based on the specified safety goals in order to identify failures that could violate these safety goals and are caused by the item under development. This analysis shall be done on each level of development that is in the scope of the project.

The next step is the specification of safety measures/safety mechanisms to avoid, mitigate, or handle the identified safety-critical failures. These safety measures/safety mechanisms should be

specified in a functional and a technical safety requirements process. Each safety-relevant requirement inherits a safety integrity level that depends on the considered system UC.

Further details about safety integrity level can be found in the following standards:

- ISO26262 for road vehicles
- ISO 25119 for Tractor,
- ISO 15998 for Earth moving machinery
- IEC 61508 for all other safety relevant systems

4.2.5 Powertrain & Vehicle Validation / Verification

Powertrain validation & verification proves the fulfillment of two main aspects:

- the satisfaction confirmation of vehicle targets and
- the satisfaction confirmation of system requirements.

Based on the system requirements and associated use cases, test cases have been defined and are evaluated to confirm that the acceptance criteria for system requirements are fulfilled.If an acceptance criterion is not fulfilled (test has failed), the sub-sequent investigation will also feedback whether the requirement or the acceptance criterion is not accurate/valid which could result in requirement/criterion changes.

However, requirement/criterion changes have to be confirmed by the project management/customer with sufficient proof that there are no detrimental effects on the system targets. Additionally, the requirement cascade has to be reviewed and evaluated for interference with other (sub-) requirements. This needs to be supported by the applied tool chain.

4.3 Main Deliverables of System Development Activities

The following list outlines the main deliverables of the systems engineering activities introduced in the V-model discussed above. The content of these deliverables have to be managed in the respective tool chain, including relevant information and interfaces that are required for the development of these deliverables.

- Use Case
  - Description of interaction between user, system, and environment
- Requirement Specification Document
  - Description / Documentation of requirements incl. rational and acceptance criteria
  - Requirements for operation states, functionality, behavior incl. transitions between modes
  - Operation & control strategy requirements
  - EE Architecture & Board Net (LV & HV) system requirements
  - Description of thermal requirements incl. rational and acceptance criteria
- Technical Specification Document
  - General component description, (structural) architecture description, description of component functionality (as part of Technical Specification Document, Component Data Sheets)
  - Functional architecture description incl. analogue / CAN interfaces as part of technical specification document
  - Interface requirements specification incl. interface description as e.g. boundary diagram

  - Description of behavior and functions incl. input, output, control & noise factors & error states
- Safety & Robustness
  - Item Definition
  - Hazard Analysis and Risk Assessment
  - (functional) FMEA
  - Functional Safety Concept (FSC)
  - Technical Safety Concept (TSC)
- Test Description
  - Description of test case, test procedure, and acceptance criteria
  - Design Verification Plan (DVP) incl. description of test methodology and relation to requirements / test cases
  - Component load cycles
- Vehicle Benchmark
  - Benchmark report, simulation model, functional description incl. re-engineering of control functionality
- Simulation / Integration
  - Function integration plan (FIP)
  - Test case simulation
- Software
  - Support of SW development in terms of MiL/SiL/HiL (Plant Model)
- Calibration
  - Calibration guidelines and acceptance criteria
- Summary Reports and Documentation
  - Review report incl. description of system functionality, behavior, components and control strategy in term of qualitative evaluation; simulation report in terms of quantitative evaluation
  - Supplier assessment report
  - Technology assessment report
  - System simulation report
- Documentation of concept for cross discipline communication incl. required exchange of information at defined points in time (Quality Gates) to ensure overall system functionality (HW+SW)
- Description of required Criteria incl. Sensitivity and Trade-Off Analysis; Confirmation of Criteria

4.4 Applied Methodologies within Development Activities

This chapter describes the main development and engineering methods identified for the scope of this use case. Each method is described in general including the description of the pre / post-conditions and the required input/output for these methods. If applicable, support methods and applied engineering tools are described as well. The methods identified are:

- Project Management
- Systems Engineering
- Requirements Engineering and Management

- System Specification & Modelling
- Verification & Validation / Test Management
- System Analysis
- System Simulation
- Safety
  - Hazard & Risk Assessment (HRA)
  - Failure Mode and Effective Analysis (FMEA)
- Configuration & Data Management
- Change & Release Management
- Collaboration
- Re-Use and Knowledge Management

This chapter is the basis for the detailed description of the engineering methods and the derivation of interoperability needs & requirements.

4.4.1 Project Management

4.4.1.1 Method Description

Project management is the discipline of planning, organizing, motivating, and controlling resources to achieve specific goals. A project is a temporary endeavor designed to produce a unique product, service or result with a defined beginning and end (usually time-constrained, and often constrained by funding or deliverables), undertaken to meet unique goals and objectives, typically to bring about beneficial change or added value. The temporary nature of projects stands in contrast with business as usual (or operations), which are repetitive, permanent, or semi-permanent functional activities to produce products or services. In practice, the management of these two systems is often quite different, and as such requires the development of distinct technical skills and management strategies.

The primary challenge of project management is to achieve all of the project goals and objectives while honoring the preconceived constraints. The primary constraints are scope, time, quality, and budget. The secondary challenge is to optimize the allocation of necessary inputs and integrate them to meet pre-defined objectives. This includes e.g. the allocation of resources to specific tasks as well as deliverables.

4.4.1.2 Input / Pre-Conditions

Main input for the project management is the contract of the development project defining e.g. goals, features/functions to be developed, components, respective restrictions for development, etc. A traditional phased approach identifies a sequence of steps to be completed. Five developmental steps of a project can be distinguished in typical development phases of an engineering project

- initiation
- planning and design
- execution and construction
- monitoring and controlling systems
- completion

4.4.1.3 Output / Post-Conditions

Not all projects follow this plan, since projects are terminated before they reach completion, some do not follow the monitoring process, and some projects will go through steps 2, 3, and 4 multiple times. Many industries use variations of these project stages. For example, when working on a brick-and-mortar design and construction, projects will typically progress through stages like pre-planning, conceptual design, schematic design, design development, construction drawings (or contract documents), and construction administration. In software development, this approach is often known as the waterfall model, i.e., one series of tasks after another in linear sequence. Waterfall development works well for small, well defined projects, but often fails in larger projects of undefined and ambiguous nature. The Cone of Uncertainty explains some of this as the planning made on the initial phase of the project suffers from a high degree of uncertainty. This becomes especially true as software development is often the realization of a new or novel product.

In projects where requirements have not been finalized and can change, requirements management is used to develop an accurate and complete definition of the behavior of software that can serve as the basis for software development. While the terms may differ from industry to industry, the actual stages typically follow common steps to problem solving—"defining the problem, weighing options, choosing a path, implementation, and evaluation."

4.4.1.4 Supporting Methods / Sub-Activities

- Project Plan, Milestones, Quality Gates & Deliverables, RASI,…
- Definition and Support of Quality and Process related Development Activities
- Definition of Process RQ's (e.g. Safety Process, Legislation,…)
- Definition of Project Boundary Conditions (Supplier, CarryOverParts, Process RQ's,…)

4.4.1.5 Applied Tools

PLM tools

4.4.2 Systems Engineering

4.4.2.1 Method Description

Systems engineering is an interdisciplinary field of engineering that focuses on how to design and manage complex engineering projects over their life cycles. Issues such as reliability, logistics, and coordination of different teams (requirements management), evaluation measurements, and other development tasks become more difficult when dealing with large, complex projects. Systems engineering deals with work-processes, optimization methods and risk management in such projects. It overlaps with technical and human-centered disciplines such as control engineering, industrial engineering, organizational studies, and project management. Systems engineering ensures that all aspects of a project or system are considered, and integrated into a whole.

4.4.2.2 Input / Pre-Conditions

As illustrated in Figure 4-7, systems engineering management is accomplished by integrating three major activities:

• *Development phasing* that controls the design process and provides baselines that coordinate design efforts,

• A *systems engineering process* that provides a structure for solving design problems and tracking requirements flow through the design effort, and

• *Life cycle integration* that involves customers in the design process and ensures that the developed system is viable throughout its lifetime.

Each of these activities is necessary for a proper management of the development effort. Phasing has two major purposes: it controls the design effort and it is the major connection between the technical management effort and the overall acquisition effort. It controls the design effort by developing design baselines that govern each level of development. It interfaces with acquisition management by providing key events in the development process, where design viability can be assessed. The viability of the developed baselines is a major input for acquisition management milestone decisions. As a result, the timing and coordination of technical development phasing and the acquisition schedule is critical in order to maintain a healthy acquisition program. The systems engineering process is the heart of systems engineering management. Its purpose is to provide a structured but flexible process that transforms requirements into specifications, architectures, and configuration baselines. Systems Engineering for this process provides the control and traceability to develop solutions that meet customer needs. The systems engineering process may be repeated one or several times during any phase of the development process. Life cycle integration is necessary to ensure that the designed solution is viable throughout the life of the system. It includes the planning associated with product and process development, as well as the integration of multiple functional concerns into the design and engineering process. In this manner, product cycle-times can be reduced, and the need for redesign and rework substantially reduced.

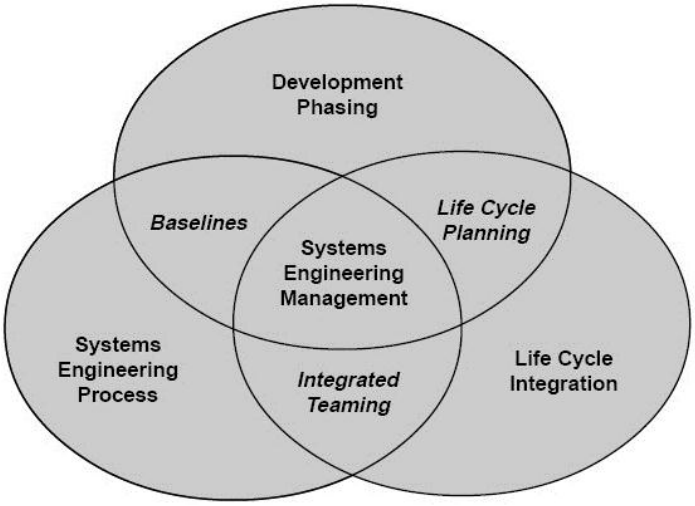

Figure 4-7: Three Activities of Systems Engineering Management [Source]

4.4.2.3 Output / Post-Conditions

The Systems Engineering process (Figure 4-8) provides an increasing level of descriptive detail of products and processes with each systems engineering process application. The output of each application is the input to the next process application.

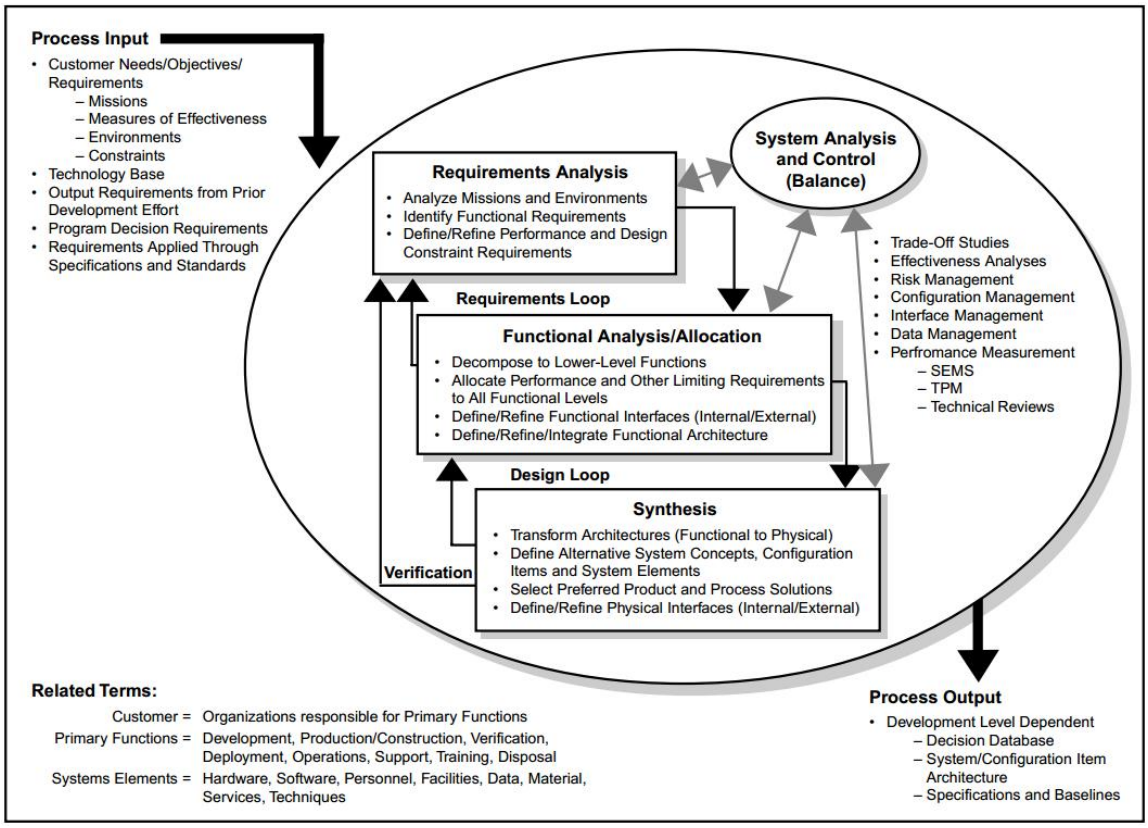

Figure 4-8: The Systems Engineering Process [Source]

4.4.2.4 Support Methods / Sub-Activities

The main sub-activities of Systems Engineering are described in the following:

Configuration management

Configuration management, as practiced in the defense and aerospace industry, is a broad systems-level practice, similar to systems engineering. While systems engineering deals with requirements development, allocation to development items, and verification, configuration management deals with the capturing of requirements, traceability to the development item, and the audit of the development item to ensure that it has achieved the desired functionality that systems engineering and/or Test and Verification Engineering have proven out through objective testing.

Use Case Definition

- Identification of Driving Cycles (e.g. NEDC, WLTP,…)
- Description of Maneuvers (incl. Mapping of System Features)
- Definition of Vehicle Use Cases incl. User / System / Environment Interaction - HMI Definition

- Lifecycle / Stakeholder Use Cases (e.g. Workshop, Shipping, Commissioning, etc.)

System Feature Development

- Definition of Operation Modes;
- Definition of System Features
- Definition of Operation Strategy

Vehicle and Powertrain Description

- Description of Energy, Material & Information Flow between main components
- Description of mechanical, thermal & functional Interfaces
- Description of EE / Communication Architecture & Interfaces
- Description of System Components (i.e. component description incl. component parameters, functionalities and interfaces)
- Description of System Features, Operation Modes, Control Strategy
- Quantitative & Qualitative Evaluation and Review of Vehicle / Powertrain Architecture
- Description of Vehicle / Powertrain Architecture

Reliability engineering

Reliability engineering is the discipline of ensuring that a system meets the customer expectations for reliability throughout its life; i.e., it does not fail more frequently than expected. Reliability engineering applies to all aspects of the system. It is closely associated with maintainability, availability, and logistics engineering. Reliability engineering is always a critical component of security and safety engineering (e.g failure modes and effects analysis (FMEA) and hazard fault tree analysis).

Support of Sub-System Specification / Development / Verification

- Support of Calibration in terms of the Development of Calibration Guidelines and Requirements
- Support of SW Development in terms of MiL/SiL/HiL / Plant Model
- Support of HW Development of components in terms of design, simulation/analysis and mechanical development
- Calibration of Hybrid System Functionality

4.4.2.5 Applied Tools

Artisan Studio

PLM / Windchill

PTC Integrity

4.4.3 Requirements Engineering & Management

4.4.3.1 Methods Description

Requirements Engineering is one of the main activities of system and software design. Requirements on one level are the basis to derive according solutions (in terms of architecture, i.e.

components & functionality) on the subsequent downstream level. This means that the developed solution satisfying the defined requirements is the basis for the derivation of requirements on a more detailed level. This is an iterative step starting at vehicle / use case level down to hardware and software component level. The methodical approach is always the same. However, different development teams with individual software tools and specific engineering methodologies are involved. Therefore different types of documentation are required:

Description of requirements across the different levels in Requirements Documents:

- Requirement Document for "***P**roduct*" – e.g. Vehicle
- Requirement Document for "***S**ystem of Product*" – e.g. Powertrain System
- Requirement Document for "***E**lement*" – e.g. ICE
- Requirement Document for "***S**ystem of Element*" – e.g. ICE Oil Circuit
- Requirement Document for "***C**omponents in Element*" – e.g. ICE Crankcase

The responsibility for the creation of these specifications depends on the project type and scope. In general, system development teams are responsible for "Product" and "System of Product", as well as partly for the "Element" description (for elements there may be an overlap with other engineering disciplines. The creation of systems and components of elements are in general within the responsibility of element teams.

4.4.3.2 Input / Pre-Conditions

Requirements Engineering & Management include activities that are relevant and have to be applied within all stages of the development process. Traditionally, Requirements Engineering focuses on the definition and development of requirements based on the customer "wants & needs" at the beginning of the development process, as well as during each development generation with increasing maturity. Requirements Management focuses on the communication of requirements between all stakeholders (customer, supplier, and internally) incl. the tracking of changes and impacts along the entire development process.

4.4.3.3 Output / Post-Conditions

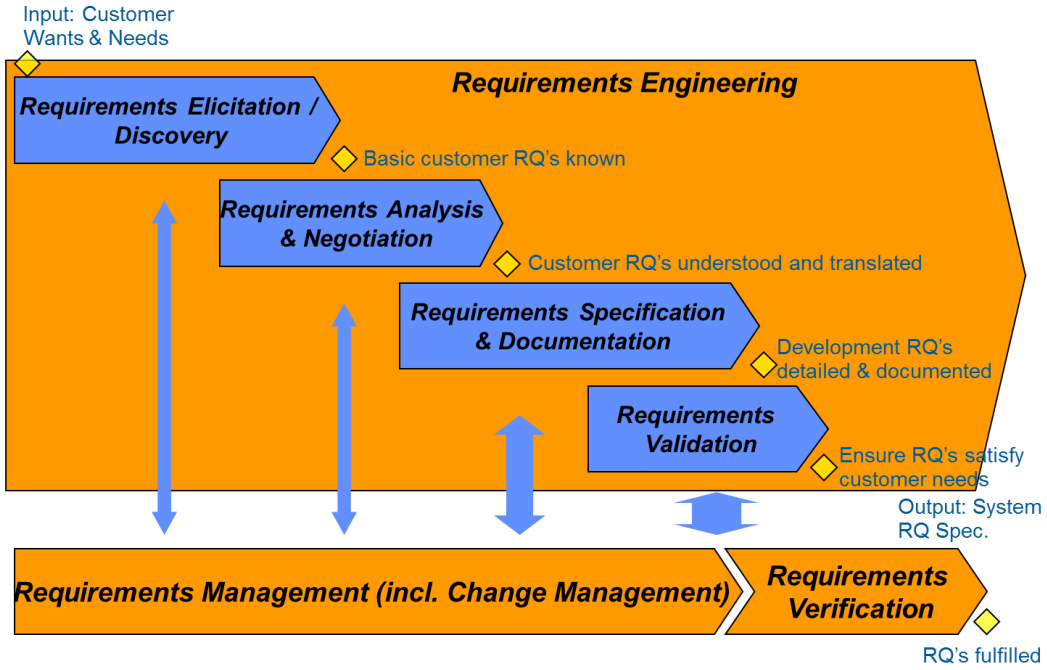

Figure 4-9: Requirements Engineering and Management Activities.

Figure 4-9 outlines the main Requirements Engineering and Management activities. The following table outlines the individual activities incl. required general input and generated output information.

Activity Name	Input	Task	Output
Requirements Elicitation / Discovery	Customer wants & needs e.g. System goals, development targets, technical and organizational boundaries (e.g. project plan), contract, benchmark, etc.	Identification and collection of relevant input that is required to start the Requirements Engineering process at project start	Collection of relevant content to serve as input to develop requirements incl. open questions to be clarified with customer – basic customer RQ's are known
Requirements Analysis & Negotiation	Collection of relevant content	Analyzing input in order to understand the meaning and purpose of the content. This includes e.g. questions, discussion, negotiations, detailing etc. in order to translate the information into "AVL language"	Structured collection of customer input incl. sources, stakeholders, and answers to open questions clarified with all stakeholders
Requirements Specification & Documentation	Structured and understood customer input	Documentation of customer input and derivation of further requirements (as the main technical activity of the Requirements Engineering process). Documentation of	All requirements are documented and described in a Requirements Management Tool (PTC Integrity) according to RQ Meta model incl. rational, acceptance criteria, and traces.

		RQ's has to be done in Requirements Management Tool (PTC Integrity)	
Requirements Validation	Intended "Solution" (e.g. specification, simulation model, etc.)	Validation in terms of checking that the system under development will result in a solution that meets the customer expectations	Statement that proposed solution will meet requirements, verified by e.g. review, simulation etc.
Requirements Verification	Developed "Solution"	Verification in terms that the "solution" fulfills the specified requirements verified through testing / V&V	Documented test results to proof that solution fulfills requirements

Table 1: Overview of Requirements Engineering Activities.

4.4.3.4 Support Methods / Sub-Activities

Requirements Engineering:

Requirements Engineering includes the process of development, cascading, and description of requirements based on initial targets (i.e. performance criteria, attributes, etc.), and features/functions to the elements / components and disciplines of the system under development. RQ's come from different sources such as e.g. OEM/customer, legal standards/regulations, and the Requirements Engineering process itself (generation of requirements). The following outlines specific examples within the requirements engineering process:

- Development of Component / Structure RQ's incl. Acceptance Criteria; (incl. Definition of Limitations, Protection Concept & Derating; Error & Fault Behavior; Safe Operation State of Components)
- Development of Feature / Function RQ's incl. Acceptance Criteria (Driving / Non-Driving);
- Development of Safety RQ's, FSC, TSC
- Development of EE Architecture & Board Net (HV & LV) Requirements incl. Power Consumption and Network Layout
- Definition of Thermal (Heating & Cooling) Requirements
- Definition of design RQ's --> Definition of geometric design modifications & boundary conditions, Requirements towards Packaging and Mounting Hints of Components within Powertrain / Vehicle
- FMEA

Requirements Management:

Requirements Management is the process of tracking and handling requirements during their lifecycle. This includes e.g. communication of requirements and changes, analysis of dependencies between requirements (traceability), impact analysis, classification, lifecycle management, etc.

Requirement Capturing / Analysis

Requirements analysis in systems engineering and software engineering encompasses tasks to determine the needs or conditions to meet for new or altered products, taking into account

possibly conflicting requirements of the various stakeholders, analyzing, documenting, validating and managing software or system requirements.

Requirements analysis is critical to the success of a systems or software project. Requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

Conceptually, requirements analysis includes three types of activities:

- Eliciting requirements
- Analyzing requirements
- Recording requirements

Eliciting requirements (e.g. the project charter or definition) means to collect business process documentation and to perform stakeholder interviews. This is sometimes also called requirements gathering.

Analyzing requirements is the activity of determining whether the stated requirements are clear, complete, consistent, and unambiguous and to resolve any apparent conflicts.

The last activity is the process of requirements recording. Requirements may be documented in various forms, usually including a summary list and may include natural-language documents, use cases, user stories, or process specifications.

Examples for vehicle requirements analysis are:

- Identification of Vehicle Targets & RQ's incl. Acceptance Criteria
- Evaluation / Definition of Environmental Conditions & Overall Boundaries for Component Operation

Requirements Allocation / Cascading

Based on functional and system analyses, system requirements must be allocated to lower-level components. The requirements allocation process is carried out iteratively in parallel with requirements analysis. Requirements allocation is a continuation of the top-down requirements engineering process.

All requirements of the top-level functions must be met by the aggregate of those for all lower level functions. Functional Analysis and Allocation of the Systems Engineering Process (see Figure 4-8) is where requirements allocation occurs.

Requirements allocation is often difficult to prove when an upper-level performance requirement is split up in a number of derived requirements. (For instance, system accuracy is composed of derived functional attributes that in sum determine its value.) Consequently, it is extremely important that higher-level requirements are allocated properly, but also that traceability to the originating requirement, and the rationale for the allocation is recorded and maintained.

Requirements Exchange

tbd

4.4.3.5 Applied Tools

PTC Integrity

SysML Tool

4.4.4 System Specification & Modeling

4.4.4.1 Method Description

The system modeling method is based on SysML and uses defined modeling elements within a specifically developed modeling methodology at AVL. These model elements are used in combination with other methods, e.g. the **P-Diagram** approach which is a tool into quality processes to achieve robustness and reliability[1].

For system modeling the P-Diagram approach is the central methodology in this chapter. The aim of a P-Diagram presentation is an integrated assessment of system functionality, its necessary inputs and desired outputs together with disturbances and (system) internal function control. As an input for the P-Diagram a Boundary Diagram needs to be depicted (see definition below).

Boundary Diagram

A Boundary Diagram helps to define the interfaces of a particular system. It provides additional information which will be useful for the analysis team when they attempt to identify potential failure modes.

Remark:
This method is also applied for safety investigations (e.g. item definition). Figure 10 shows an example boundary diagram for an item under investigation. Each component within the boundaries is in the scope of the investigated functional safety concept. All components outside the boundaries are interfacing directly the item under investigation. Those interfacing components are not investigated directly. Only if there are requirements connecting the item and these components, they will be mentioned in the diagram and the requirements will be listed. The responsible partner for this component has to assure the implementation in the demanded quality.

[1] Robustness can be described as a distance between a system output and an applied load of a system function. Reliability is the probability to achieve a system function over a period of time, kilometer range, or cycle number including external disturbance.

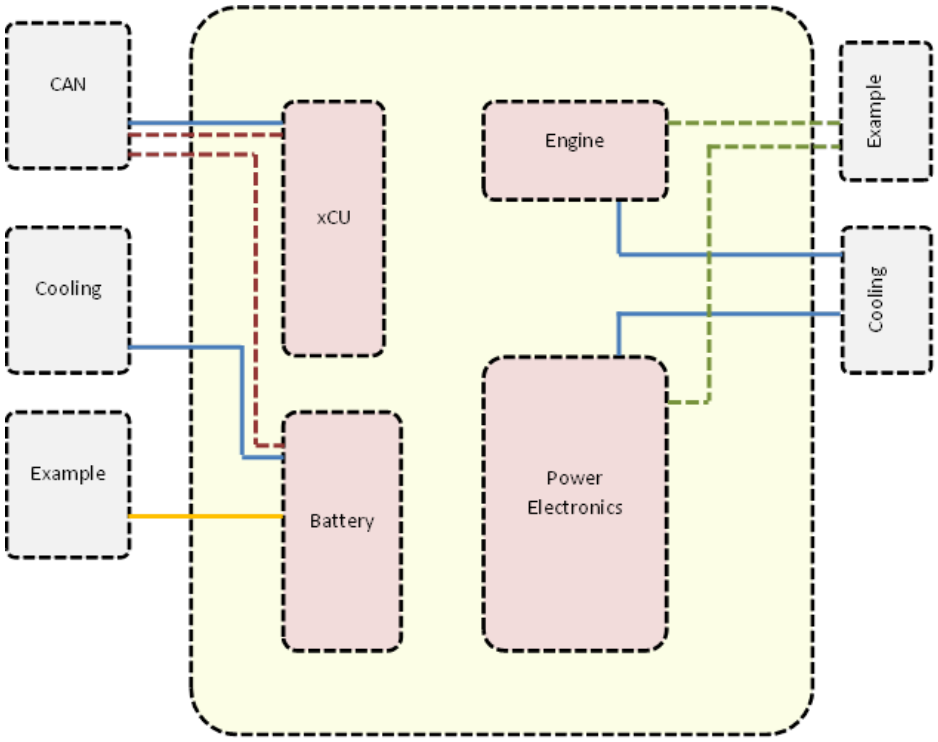

Figure 10: Boundary Diagram Structure

4.4.4.2 Input / Pre-Conditions

For the creation of a P-Diagram during system modeling following tools and roles are useful:

- A block diagram (also boundary diagram) that describes the influences inside and outside of the system.
- A mind map has been shown to be useful especially in the consideration of functions.
- It is advantageous that a moderator leads the P-Diagram creation.
- An expert team that is intimately familiar with the current theme should not be missing.
- Functional Description

4.4.4.3 Output / Post-Conditions

As shown in the Figure 11 the center of the P-diagram is reserved for the system/ process/ product/ function (grey box). The initial ideal output(s) (purple box, right) is/are defined as a physical value including the inevitable losses. With the ideal output, it can be easier to determine the input (purple box, left). It is possible to use physical values.

- It also has to be considered that no noise factors (see below) are described as an input.

The undesired effects (yellow box) should be balanced in terms of users and the technical approach.

- User perspective is all that can be detected by our senses.

The noise factors (red boxes) are fragmented into 5 groups:

- Total Design/ Manufacturing variability
- Component changes over time / mileage
- Customer usage / duty cycle

- External environment
- In-Vehicle system environment

The control factors (green boxes) used to eliminate or to counteract the noise factors and undesired effects. The Control Factors are fragmented into three subgroups:

- specifications
- calibration
- and software

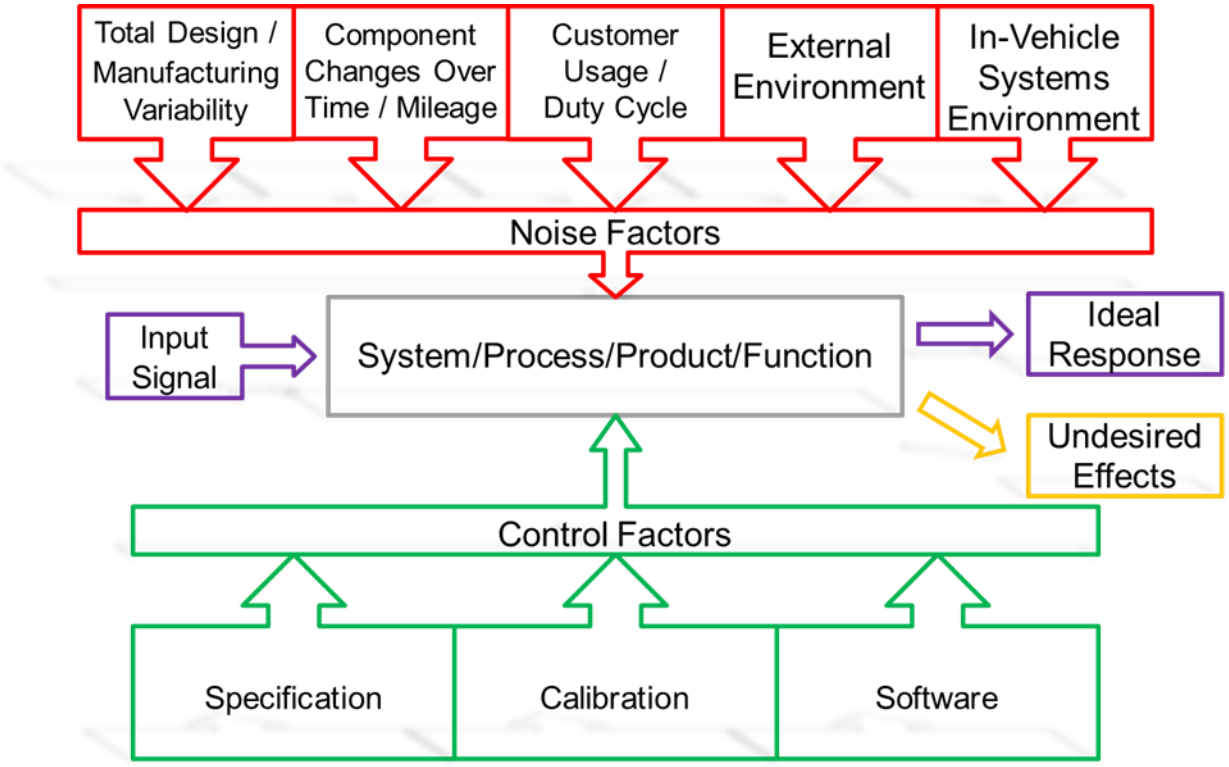

Figure 11: P-Diagram Overview

4.4.4.4 Support Methods / Sub-Activities

P-Diagrams can be used to identify additional specifications, calibration tasks, and software functions. In addition, they can be used as input for an FMEA or as a helpful document in order to derive detailed requirements.

4.4.4.5 Applied Tools

To develop a P-Diagram, Excel Files with Macros or a system modeling tool supporting SysML can be used.

4.4.5 Verification & Validation / Test Management

4.4.5.1 Method Description

System validation checks that the product design satisfies or fits the intended use (high-level checking), i.e., the software meets the user requirements. This is done through dynamic testing

and other approaches such as long-term static tests. Verification and validation tackle different aspects.

"Validation: The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers. Contrast with verification."

"Verification: The evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process. Contrast with validation."

In other words, validation ensures that the product actually meets the user's needs, and that the specifications were correct in the first place, while verification ensures that the product has been built according to the requirements and design specifications.

4.4.5.2 Input / Pre-Conditions

An input for the Validation and Verification Process can be the definition of “Test Cases” including the acceptance criteria or test objectives to verify the requirements and use cases. Other inputs are the description of the test procedure including test conditions and the environment. Input for test cases are use cases as well as the variation space defined though a set of component parameters / control factors which map the specified requirements.

4.4.5.3 Output / Post-Conditions

The output generated by validation is the knowledge that the product actually meets the user's needs, and that the specifications were correct in the first place, while verification is ensuring that the product has been built according to the requirements and design specifications.

4.4.5.4 Support Methods / Sub-Activities

A **Test Case** is a method used in the process. Test Cases may be prepared for verification - to determine if the process that was followed to develop the final product is right -, and also for validation - if the product is built according to the requirements of the user. Other methods, such as reviews, may be used early in the life cycle to support validation. Next to Test Cases, some examples for support methods during Verification & Validation are:

- Verification of Requirements (verify RQ acceptance criteria)
- Definition and Monitoring of Functional Design Verification & Test Planning --> DVP (Design Verification Plan)
- Evaluation and Simulation of Test Cases based on e.g. Driving Maneuvers, Safety Cases, etc.
- Target Verification on Powertrain / Vehicle Test Bed (e.g. PT Test Bed, Chassis Dyno, Vehicle Testing)

4.4.5.5 Applied Tools

SysML Tool, PTC Integrity, In Motion - Test Manager

4.4.6 System Analysis

4.4.6.1 Method Description

Analysis is this context is the qualitative investigation of a complex system into smaller parts to gain a better understanding of it. Analysts in the field of engineering look at requirements, structures, mechanisms, functionalities and additional dimensions required for investigations. The main focus of analysis is the quantitative evaluation of the system model, i.e. it must be assured that the system model includes all relevant aspects and answers questions such as “are all component interfaces linked to functional control factors” etc.

4.4.6.2 Input / Pre-Conditions

As input for analysis methods, the processed and consolidated knowledge of the system and the identification of structure and functionalities of existing systems are needed. Boundary diagrams support the identification of operations. This is the first step towards re-engineering out of system measurements or paper/descriptions.

4.4.6.3 Output / Post-Conditions

The output is a better understanding of the system or the problem at hand. A good example for an output of an analysis is the form sheet of a Failure Mode and Effects Analysis.

4.4.6.4 Support Methods / Sub-Activities

Tbd

4.4.6.5 Applied Tools

SysML Tool

4.4.7 System Simulation

4.4.7.1 Method Description

System simulation is a set of techniques that use computers to imitate the operations of various real-world tasks or processes through simulation. Computers are used to generate numeric models for the purpose of describing or displaying complex interactions between multiple variables within a system. The complexity of the system arises from the stochastic (probabilistic) nature of the events, rules for the interaction of the elements, and the difficulty in perceiving the behavior of the systems as a whole with the passing of time.

4.4.7.2 Input / Pre-Conditions

The input of System Simulations can be the parameters of the system based on the confirmed system targets, as well as former simulation results.

4.4.7.3 Output / Post-Conditions

The motivation for System Simulation is based on the following output:

- Reduction of development time and costs
- Assistance for control algorithm development
- Assistance for calibration
- Assistance for testing

- Frontloading – no need of hardware of e.g. engines, transmissions or vehicles in early development phases
- Independence to mechanical development progress
- Gaining system experience without real system
- Reproducibility of physical system behavior
- Support for investigations of suspect system behavior
- Generation of functional requirements
- Reuse of the plant models in follow-up projects Function and Software Development Process Simulation Method Description.
- Supporting function and element/component specification
- Supporting SW development (Plant Modeling)
- Supporting of system calibration (pre-calibration of e.g. energy management)

4.4.7.4 Support Methods / Sub-Activities

Plant Models

The description of the physical plant you want to be controlled, including driver, road, etc.

System Target Analysis and Confirmation

- Definition and Evaluation of Component Load Collective based on Driving Cycles (Certification / Real World)
- Vehicle / Powertrain Simulation / Confirmation of Targets / Attributes

4.4.7.5 Applied Tools

CRUISE[2], InMotion, AMESim, Matlab,…

4.4.8 Safety – Hazard and Risk Assessment (HRA)

4.4.8.1 Method Description

The purpose of the HRA is to create an adequate understanding of the hazards and risks concerning an item, i.e. the system under investigation. The primary objective is to identify and classify the hazards that may arise due to potential faulty behavior during the operation of an item. The item, its dependencies on, and interaction with the environment and other items shall be described and analyzed. The classification is done according to the ISO26262 and includes critically levels from ASIL A (Automotive Safety Integrity Level) to ASIL D, as well as QM (Quality Management) ratings for errors not rated to be safety relevant. The focus is set to systems, listed in an item definition document. The objective of the Hazard and Risk Analysis is:

- to identify the potential hazards of the item,
- to categorize these hazards and,
- to formulate the safety goals related to the prevention or mitigation of these hazards.

[2] See also Deliverable D603.011 for details concerning AVL CRUISE

4.4.8.2 Input / Pre-Conditions

The HRA is based on an Item Definition document as a special view of System Specification and Modelling. The creation method is divided in 2 main phases, as shown on Figure 4-12..

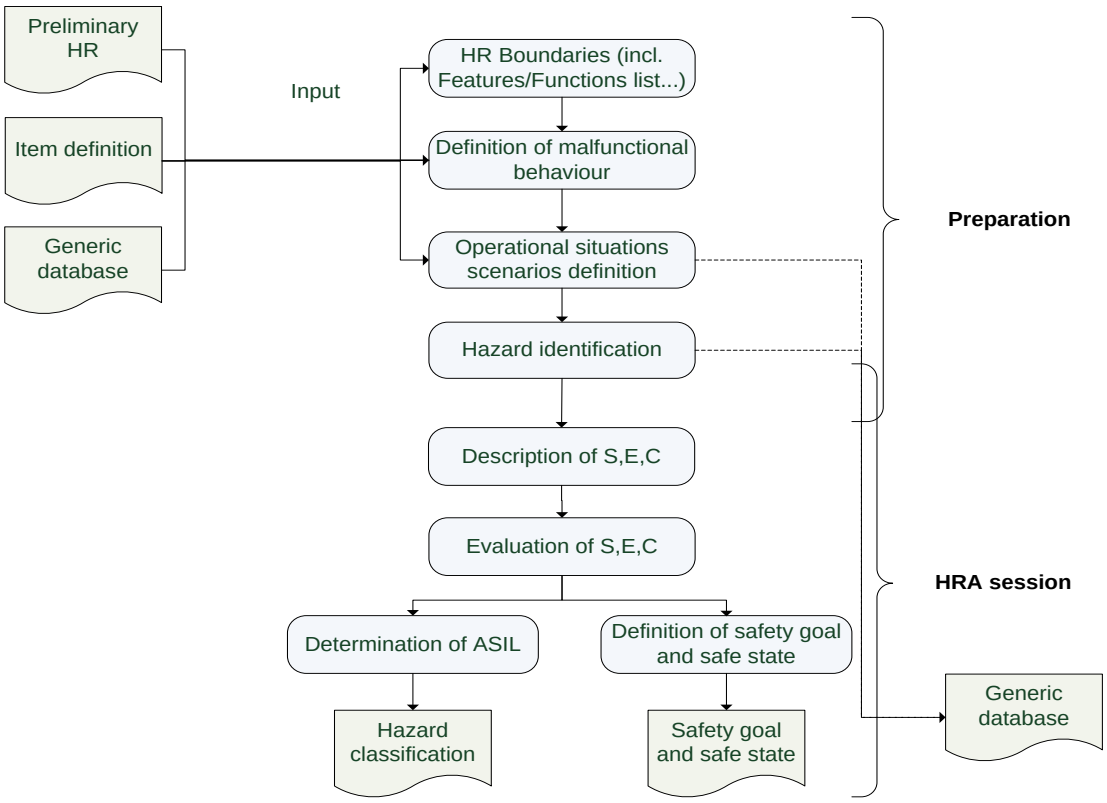

Figure 4-12 Creation method HRA

4.4.8.3 Output / Post-Conditions

List of identified Hazards and Risks

The HRA is displayed as a Table, consisting of 19 columns.

Hazard ID	**Function // System Component** (of preliminary architecture)	**Modes // Sub-system Components**	**Guideword**	**Hazard Code**	**Malfunction/ Failure mode**	**Situations/ Modes**	**Example**

...

Hazard Impact (at vehicle level)	**S**	**Severity /** Possible Consequences	**E**	**Exposure /** Affected situations / use cases	**C**	**Controllability /** assumptions, driver tests, etc.

...

ASIL	**Safety Goal Code**	**Safety Goal**	**Possible countermeasures**

Figure 4-13: Layout of the HRA table

Malfunctional Behavior

Hazards are specified by combining a potential malfunction and a concrete operational situation of the item. This is usually done in a HRA session or workshop. Malfunctions can be identified by using the following Hazop study guidewords in relation to each function:

- No
- Unintended
- More
- Less
- As Well As
- Part Of
- Reverse
- Other Than
- Intermittent
- Early
- Late
- Before
- After

Operational Situations

If the item under consideration is a vehicle, a hazardous situation would involve persons such as vehicle occupants (including driver), other road users (occupants of other vehicles as well as cyclists), or nearby pedestrians, and mechanics (in the workshop). For the vehicle, operational conditions are identified as combination of the four following parameters:

Road conditions	Traffic situations	Drive situations	Environment / specific description of the environment
E.g. highways, main roads or city areas.	E.g. no traffic, heavy traffic, lane change, trailer operation	E.g. high velocity, intermediate speed or creep.	E.g. normal, black ice or oncoming traffic

Table 2: Operational Situations

Hazard classification

The identified hazards are classified regarding their Severity, Exposure, and Controllability:

- **Severity (S):** evaluation of hazard consequences in terms of potential harm to persons caused by malfunctions of a system,
- **Exposure (E):** evaluation of the relative frequency or duration of relevant use conditions when the hazard occurrence may lead to harm,
- **Controllability (C):** evaluation of the probability that the harm cannot be avoided or mitigated by human intervention in the given situation.

Severity Table
Exposure Table
Controllability Table
Determination of the ASIL
Definition of safety goal and safe state

4.4.8.4 Support Methods / Sub-Activities

4.4.8.5 Applied Tools

SysML Tool incl. Safety Extensions, IQ-FMEA, Fault Tree Generator, etc

4.4.9 Safety – Failure Mode and Effects Analysis (FMEA)

4.4.9.1 Method Description

An FMEA is often the first step of a system reliability study. It involves reviewing as many components, assemblies, and subsystems as possible to identify failure modes and their causes and effects. For each component, the failure modes and their effects on the residual system are recorded in a specific FMEA Form Sheet. A FMEA is mainly a qualitative analysis and different types of FMEAs exist, e.g. Functional, Design, and Process FMEA.

An FMEA activity helps to identify potential failure modes based on experience with similar products and processes - or based on common physics of failure logic. It is widely used in development and manufacturing industries in various phases of the product life cycle. Effects analysis refers to studying the consequences of those failures on different system levels. The FMEA focuses on measures to be considered during component specifications, system design, and functional development of an item to prevent safety hazards and quality issues. The FMEA is created according to the safety development process as defined in ISO 26262. The components on system and sub-system level have to be identified and functions and failure modes have to be analyzed. Each failure mode is rated and cross checked with the Hazard and Risk Analysis.

4.4.9.2 Input / Pre-Conditions

Input for the FMEA is e.g. a system specification / item definition including description of components & behavior such as functionalities, operation modes etc.

4.4.9.3 Output / Post-Conditions

For the creation of the System FMEA, the item is fragmented into its systems and components (see the diagram below). The System FMEA is applied to all (sub-) systems on system level. The FMEA structure explained is an example and based on a vehicle topology.

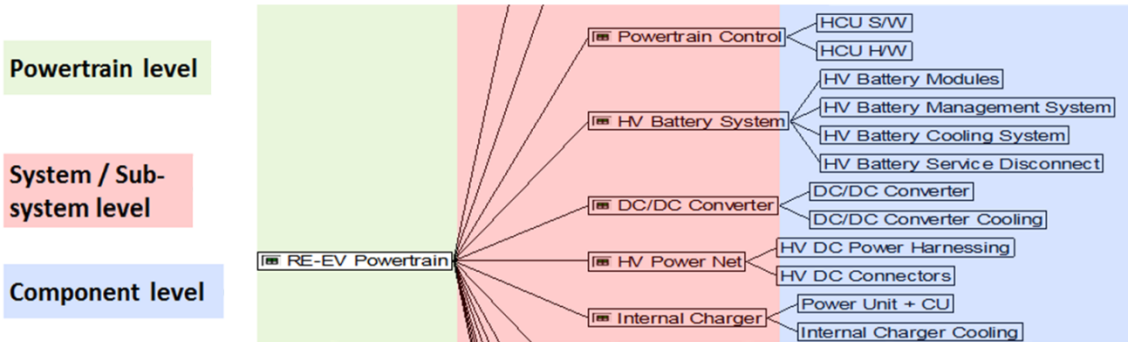

Figure 4-14: System FMEA structure

As a first step of the creation of a System FMEA Form Sheet, the architecture of the system is defined and system elements and component elements are identified and linked to each other. Afterwards, the functionalities and failure modes on system level are identified and linked on

powertrain level as failure effects and on component level as failure root cause. Dependent on the rating of the severity of a failure effect on powertrain level, countermeasures in terms of preventive or detection actions are identified on component level.

Rules considered during System FMEA creation

The System FMEA is created according to the following rules:

- Failure-modes refer only to the respective function
- Only single failures are considered
- Failure-effects have an influence on the system, on other systems, or on the whole car and / or its passengers
- Only failure-causes leading to a specific system failure mode are analyzed, if they emerge in the same system in which the failure mode occurs.
- Preventive actions are defined to avoid the failure-cause, however, if the cause cannot be prevented in some cases they are defined to prevent the failure-effect
- Detection actions are defined to determine failure-causes as soon as possible, without preventing the cause. In rare cases they detect failure-modes or failure-effects.

System FMEA creation steps

Based on the structure above, the following steps are performed:

1. The main functions for all systems are identified.
2. For each system and each related function, potential failure modes are identified which compromise the functionality of the system or the safety of persons.
3. For each system failure mode, potential effects on “powertrain level” are identified and rated concerning their severity.
4. For each system failure mode, potential causes on component level are identified.
5. For each failure mode on component level causing effects with high severity (≥ 7), preventive actions and detection actions are defined.

The process is also described in the following example chart, taken from the FMEA form sheets, created at the end of the FMEA development process:

Usage Options and further steps

Additional safety related requirements are defined in the Functional Safety Concept (FSC), which needs also to be considered for system and component design. After rating the severity, occurrence, and detection for each failure mode, a risk and priority number (RPN) is calculated. This risk and priority number is used to identify critical failures and to decide if further measures for failure prevention need to be taken or not.

The System FMEA generates output to create Component FMEAs for each component based on the functions and failure modes identified on component level. The creation of a Component FMEA is done by the respective suppliers of the component.

4.4.9.4 Support Methods

The FMEA is supported by the Hazard and Risk Analysis and the Functional Safety Concept.

4.4.9.5 Applied Tools

SysML Tool incl. Safety extension, APIS IQ-FMEA ,…

4.4.10 Configuration & Data Management

4.4.10.1 Method Description

Configuration and Data Management is a business function often within product lifecycle management that is responsible for the management and publication of product data. In software engineering, this is known as version control. To manage product data, software or other tools are used to track and control data related to a particular product. It is also part of product lifecycle management and configuration management, and is primarily used by engineers.

4.4.10.2 Input / Pre-Conditions

The data tracked usually involves the technical specifications of the product, specifications for manufacture and development, and the types of materials that will be required to produce goods. The use of product data management allows a company to track the various costs associated with the creation and launch of a product.

Typical information managed in the Data Management module includes:

- Part number
- Part description
- Supplier/vendor
- Vendor part number and description
- Unit of measure
- Cost/price
- Schematic or CAD drawing
- Material data sheets

4.4.10.3 Output / Post-Conditions

Within data management the focus is on managing and tracking the creation, change, and archive of all information related to a product. The information being stored and managed (on one or more file servers) will include engineering data such as computer-aided design (CAD) models or drawings and their associated documents and serves as a central knowledge repository for process and product history. It promotes the integration and data exchange among all business users who interact with products — including project managers, engineers, sales people, buyers, and quality assurance teams. The central database will also manage metadata such as owner of a file and release status of the components. The package will: control check-in and check-out of the product data to multiple users; carry out engineering change management and release control on all versions/issues of components in a product, and assist in configurations management of product variants.

4.4.10.4 Support Methods

In industry, **product lifecycle management (PLM)** is the process of managing the entire lifecycle of a product from inception, through design and manufacture, to service and disposal. PLM integrates people, data, processes and business systems and provides a product information backbone for companies and their extended enterprise.

4.4.10.5 Applied Tools

Tools supporting the Systems Modeling Language (SysML), a modeling language used for systems engineering applications, supports the specification, analysis, design, verification and validation of a broad range of complex systems and can be used to handle data management during an engineering project. Other Tools used are *Integrity* and *Windchill.*

4.4.11 Change & Release Management

4.4.11.1 Method Description

Change and release management is the process of managing software releases from development stage to software release. It is a relatively new but rapidly growing discipline within software engineering. As software systems, software development processes, and resources become more distributed, they invariably become more specialized and complex. Furthermore, software products (especially web applications) are typically in an ongoing cycle of development, testing, and release. Add to this an evolution and growing complexity of the platforms on which these systems run, and it becomes clear there are a lot of moving pieces that must fit together seamlessly to guarantee the success and long-term value of a product or project.

4.4.11.2 Input / Pre-Conditions

Some of the challenges of Release Management include the management of:

- Software defects
- Issues
- Risks
- Software change requests
- New development requests (additional features and functions)
- Deployment and packaging
- New development tasks

4.4.11.3 Support Methods

tbd

4.4.11.4 Applied Tools

Similar to the Data Management method i.e. Release Management Tool, SysML Tool, Integrity or PDM

4.4.12 Collaboration

4.4.12.1 Method Description

Collaboration is working with each other to do a task and to achieve shared goals. It is a recursive process where two or more people or organizations work together to realize shared goals. Collaboration projects can be structured by dividing a project into work packages. Each organization will work on a work package according its expertise to achieve the shared goals, or requirements, formulated in a contract.

4.4.12.2 Input / Pre-Conditions

Due to the complexity of today's business environment, collaboration in technology encompasses a broad range of tools that enable groups of people to work together including social networking, instant messaging, team spaces, web sharing, audio conferencing, video, and telephony. Broadly defined, any technology that facilitates linking of two or more humans to work together can be considered a collaborative tool. Wikipedia, Blogs and even Twitter are collaborative tools. Many large companies are developing enterprise collaboration strategies and standardizing on a collaboration platform to allow their employees, customers and partners to intelligently connect and interact. The inputs of collaboration are included in a contract and individual needs, depending on the project at hand.

4.4.12.3 Output / Post-Conditions

The output is a successfully project with completed work packages, reviewed by all collaboration partners.

4.4.12.4 Support Methods

Enterprise collaboration tools are centered around attaining collective intelligence and staff collaboration at the organization level, or with partners. These include features such as staff networking, expert recommendations, information sharing, expertise location, peer feedback, and real-time collaboration. At the personal level, this enables employees to enhance social awareness and their profiles and interactions Collaboration encompasses both asynchronous and synchronous methods of communication and serves as an umbrella term for a wide variety of software packages.

4.4.12.5 Applied Tools

Perhaps the most commonly associated form of synchronous collaboration is web conferencing using tools like IP telephony, instant messaging, and rich video interaction with telepresence. Examples of asynchronous collaboration software include Cisco WebEx Connect, GoToMeeting or Microsoft SharePoint, MS-Lync.

4.4.13 Re-Use and Knowledge Management

4.4.13.1 Method Description

Re-Use and Knowledge Management is about making the right knowledge available to the right people and making sure that an organization can learn, and that it will be able to retrieve and use its knowledge assets in current applications as they are needed. Knowledge management is the systematic management of an organization's knowledge assets for the purpose of creating value and meeting tactical & strategic requirements. A number of claims exist as to the motivations leading organizations to undertake a Knowledge Management effort. Typical considerations driving a Knowledge Management effort include:

- Making available increased knowledge content in the development and provision of products and services
- Achieving shorter new product development cycles
- Facilitating and managing innovation and organizational learning
- Leveraging the expertise of people across the organization
- Increasing network connectivity between internal and external individuals

- Managing business environments and allowing employees to obtain relevant insights and ideas appropriate to their work
- Solving intractable or wicked problems
- Managing intellectual capital and intellectual assets in the workforce (such as the expertise and know-how possessed by key individuals)

4.4.13.2 Input / Pre-Conditions

It consists of the initiatives, processes, strategies, and systems that sustain and enhance the storage, assessment, sharing, refinement, and creation of knowledge. Many large companies and non-profit organizations have resources dedicated to internal Knowledge Management efforts, often as a part of their business strategy, information technology, or human resource management departments. Several consulting companies also exist that provide strategy and advice regarding KM to these organizations.

4.4.13.3 Output / Post-Conditions

Knowledge management implies a strong tie to organizational goals and strategy, and it involves the management of knowledge that is useful for some purpose and which creates value for the organization. The output is therefore to create/provide the right tools, people, knowledge, structures (teams, etc.), culture, etc. so as to enhance learning; it shall understand the value and applications of the new knowledge created and store this knowledge and make it readily available for the right people at the right time. It shall also continuously assess, apply, refine, and remove organizational knowledge in conjunction with concrete long and short term factors.

4.4.13.4 Support Methods

Knowledge engineering is an engineering discipline that involves integrating knowledge into computer systems in order to solve complex problems normally requiring a high level of human expertise.

4.4.13.5 Applied Tools

tbd

5 Detailed Engineering Method Descriptions

5.1 Requirements engineering using Controlled Natural Language

5.1.1 General description

Controlled Natural Language (CNL) is a subset of natural language that has a restricted grammar and vocabulary. It can be used to bridge the gap between natural language and formal languages. Boilerplates are a popular way of using CNL for requirements. Boilerplates are predefined sentence templates (fixed structure) that contain variable parts (placeholders) that have to be filled by the requirements engineer. The content of these variable parts can be restricted by using only language elements which have been defined in a glossary. Figure 5-1 shows an example for a boilerplate and a corresponding boilerplate based requirement.

Boilerplate:
The <system> shall <function> every <quantity> <unit>.

Corresponding requirement:
The *HCU* shall *check SOC* every *1 ms*.

Figure 5-1: Example for boilerplate requirement

The application of boilerplates for requirements engineering is a good instrument to improve the quality of requirements. As the requirements are specified consistently with defined terms, the ambiguity of requirements can be reduced. Further, the semi-formal notation of requirements makes it possible to automate some processing steps. More information concerning boilerplates and how to use it in Crystal are described in D603.011.

5.1.2 Application of CNL

In practice there is often a gap between customer requirements and (internal) technical system requirements from the product developer. This means that the customer requirements cannot be used as single basis to develop the product. First, requirements from different stakeholders (customer, marketing, regulations, etc.) have to be analyzed, harmonized, and translated into technical system requirements which should be understood unambiguously in the company.

In Crystal, we want to find a restricted language that can be used in the automotive domain for specifying various kinds of requirements in order to support the requirements engineer in writing technical system requirements in a consistent way. Furthermore, the restricted language should help to transfer the requirements into the system design by mapping the requirements into SysML model elements. A prototype implementation should give the proof of our concept and support the requirements engineer in using the restricted language.

The restricted requirements language could support developers in mainly two process activities (Cp. Figure 5-2):

1. Analyze customer requirements and translate them into technical system requirements
2. Develop model-based system specification based on requirements

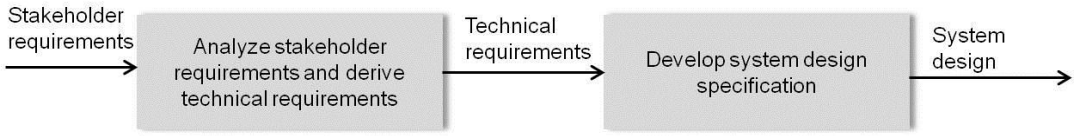

Figure 5-2: Process activities supported with CNL

5.1.3 Integration into Use Case

For the integration into the Use Case the Requirements Management tool has to cooperate with the CNL tool. A possible scenario for using CNL in the development process could be:

1. Requirements Engineer elicits requirements and writes NL requirements in Requirements Management (RM) Tool (E.g. PTC integrity)
2. RM Tool provides NL requirements in specified format (IOS)
3. Requirements engineer imports NL requirements in CNL formalization tool
4. Requirements engineer uses the CNL for developing technical system requirements
5. CNL formalization tool provides semi-formal CNL requirements with link to NL requirements in specified format (IOS)
6. Requirements Engineer exports design elements generated out of CNL requirements to Design specification (SysML) Tool
7. Requirements Engineer exports CNL requirements to RM Tool
8. RM tool saves CNL requirements and links to both NL requirements and System design elements

Figure 5-3 shows an overview of the tools that have to be integrated for deriving semi-formal technical system requirements.

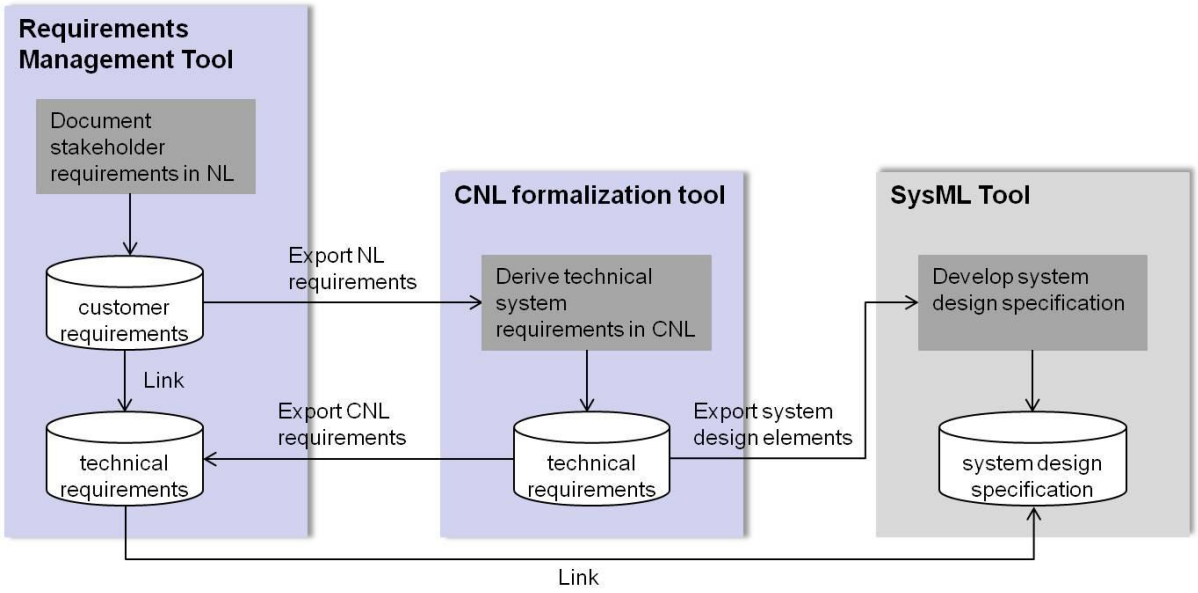

Figure 5-3: Integration of tools for deriving CNL requirements

Input for CNL tool

- Natural language requirements in defined format

Output:

- CNL requirements in defined format
- 'Derive' Link for NL and CNL requirements
- System design elements (in SysML)
- 'Satisfy' Link for CNL requirements and generated System design elements

Preconditions

- RM Tool has to provide appropriate interface for exchanging requirements
- RM Tool has to provide possibility to store additional information for CNL requirements
- Design specification tool has to provide interface

A second scenario is possible where a more integrated functionality with RM tool is necessary. In contrast to the first scenario, in this scenario single requirements are semi-formalized:

1. Requirements Engineer elicits requirements and writes NL requirements in RM Tool
2. Requirements Engineer opens function in RM Tool for deriving a semi-formal requirement
3. CNL tool opens
4. Requirements engineer uses CNL for developing technical system requirements out of the NL requirement
5. RM tool saves CNL requirement and link

This scenario could be appropriate in case a new NL requirement is inserted to the specification and the technical system requirements have to be added.

5.2 Linkage of requirements and Product Structure - PTC Integrity / Windchill integration

This subsection describes a detailed method description of a PTC Integrity / PTC Windchill integration in terms of requirements exchange and its linkage to the BoM (bill of material). This description is partly based on findings of [Rosenberger & Denger 2009].

5.2.1 Definitions

Before going into details, some engineering environment related terms should be introduced and briefly explained with respect to the intended activities.

Model

A CAD Model is a geometric description of a product. Structural models are organized as assemblies. As an extract of this structure, a BoM is created. In [Eigner 2005] the BoM was defined as a representation of the product structure, or a specific domain dependent view of the product (e.g. design or manufacturing BoM)

Component

Components in a product structure are elements that are not dividable into further sub components. They can be considered as smallest structural units within product structure.

Product

A hierarchical collection of required components defines a product.

Structure

A product as a system can be divided into subsystems [Haberfellner et al. 2002, Pahl et al. 2007]. Therefore a structure is a hierarchical subdivision of a set of elements. For instance, depending on the point of view, an engine can either be a product or a component of a car. Structures are used to form BoM's or to organize a product for different points of view.

5.2.2 Intended scenario

5.2.2.1 Challenges and issues

One main challenge today is that the architecture of the PDM-System becomes more and more complex. Current PDM-Systems are based on complex and inflexible relational database models. Usually these complex relationships between tables are interpreted by a server backend or a user application. Extending functionality or creating merged domain views with other tools requires additional implementation efforts: tables in the database have to be modified and software has to be customized each time newly.

Depending on their position in the product lifecycle, engineers have different understandings of product structures. Therefore, it is common to have several parallel structures of one product. The functional structure usually depends on requirements and the product structure on a certain view. Therefore, the mapping of requirements and product structures offers a special view on demands for products functionality and allows the engineer to verify the correctness of the functionality of this product.

5.2.2.2 Scenario

As stated in Section 5.2.1, the product structures of current systems are very rigid concerning the creation of customized views on product aspects. Because of this, an alternative approach is required to support the handling of multiple views of structures when two or more tools, each of them delivering one part of the view, should be merged. Figure 5-1 depicts an example for such a scenario. In our sample scenario two tools, one of them containing information about requirements (PTC Integrity) and the other about product structure (PTC Windchill), should be used as an experimental environment to tackle the defined problems and offer solutions and methodologies that can be applied in every day work.

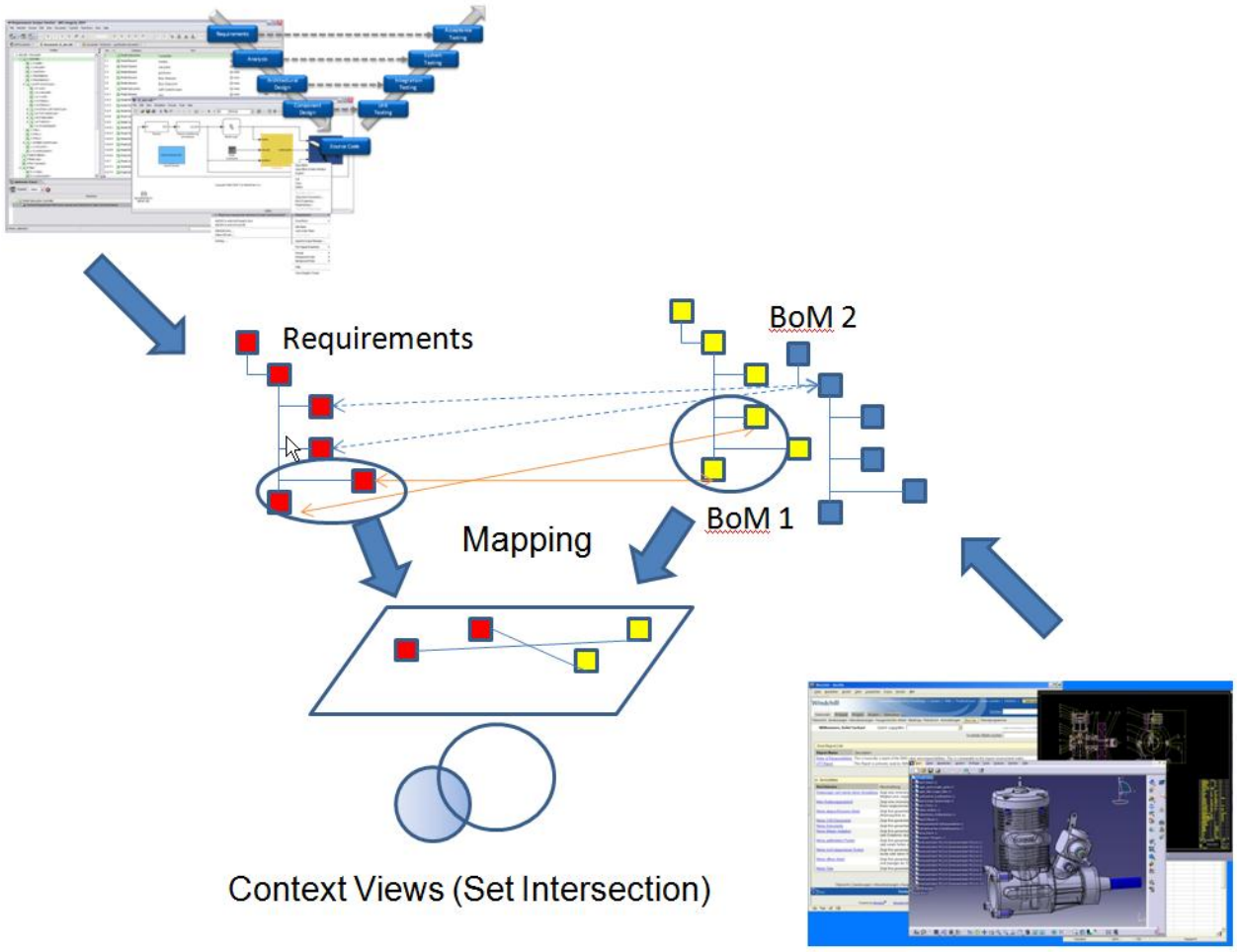

Figure 5-1: Concept of mapping requirements to the BoM

5.2.2.3 Benefits of proposed approach

The main benefit of the proposed approach is that it enables different tools to generate context-specific views on information that reflects for instance the function structure of a part or a vehicle. This enables decision support based on criteria such as costs, security, organization, etc. The most common case in practice is to generate function-related views on the tasks in order to complete the development of the product. In this way it will be possible to e.g. improve the Change Management and verify the requirements against product parts of BoMs over the functional relation.

5.2.2.4 Applied Tools

This section provides a short description of the tools intended to be used by the experimental scenario.

5.2.2.5 PTC Integrity[3],[4]

MKS Integrity is a PTC product and allows software development teams to track all aspects of their work, including work items, source control, reporting, and build management in a single tool. The product consists of two components - MKS Source and MKS Integrity. The source control part of MKS is based on a client-server architecture. The other component (MKS Integrity) is an Issue tracking system. MKS Integrity is based on a single repository. This single-repository solution supports the three pillars of lifecycle management — traceability, process automation, and reporting and analytics. Integration of MKS Integrity with IDEs and other development tools is limited to few products. Supported IDEs include Eclipse and Visual Studio. Also supported is IBM. There also exists support for MKS in Apache Maven.

5.2.2.6 PTC Windchill[5]

Windchill is a PLM (Product Lifecycle Management) software product that is offered by PTC. It was designed to support distributed product development with a Web-based architecture that can coordinate replicated databases around the world.

5.2.3 Intended possible Approaches

The standard relation type in product structures is a hierarchical relationship of parent and child nodes. This relation type is the base relation type for hierarchical structures in current PDM-Systems. However, this approach does not satisfy the requirement to create relations between two different structure views of one system from two different tools. Because of this, a different approach is required to support handling of multiple views of structures [Schuh 2005].

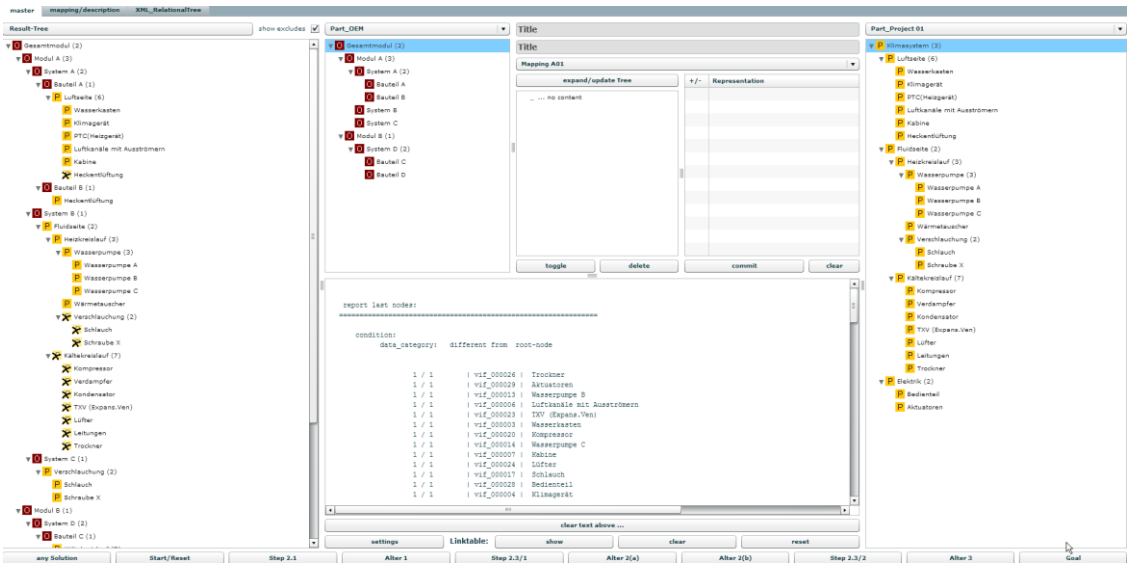

Figure 5-2: Ontology mapping of product structures

[3] http://www.mks.com/platform/

[4] http://en.wikipedia.org/wiki/MKS_Integrity

[5] http://www.inneo.co.uk/fileadmin/inneo/products/windchill/pdmlink/windchill-pdmlink-datasheet.pdf

An approach for ontology-based structure mapping seems to be a reasonable means to handle multiple views on product structures. Ontologies are a formal description of objects, their behavior, relationships, and restrictions build up as links and nodes [Hitzler et al 2008]. Therefore, they go beyond relational data storage systems. Relations as links between objects are the essential information. Rules on this links define the behavior of the ontology. This required underlying definition is a subset of the stored data [Rosenberger & Denger 2009]. Therefore modification is more flexible than in relational database systems. An experimental solution based upon ontology mapping of product structures has been proposed in [Rosenberger & Denger 2009] and is shown in Figure 5-2.

One alternative to the ontology approach is the use of an integrative solution as offered by PTC itself (see Figure 5-3.). This solution is well integrated; however, only for PTC products.

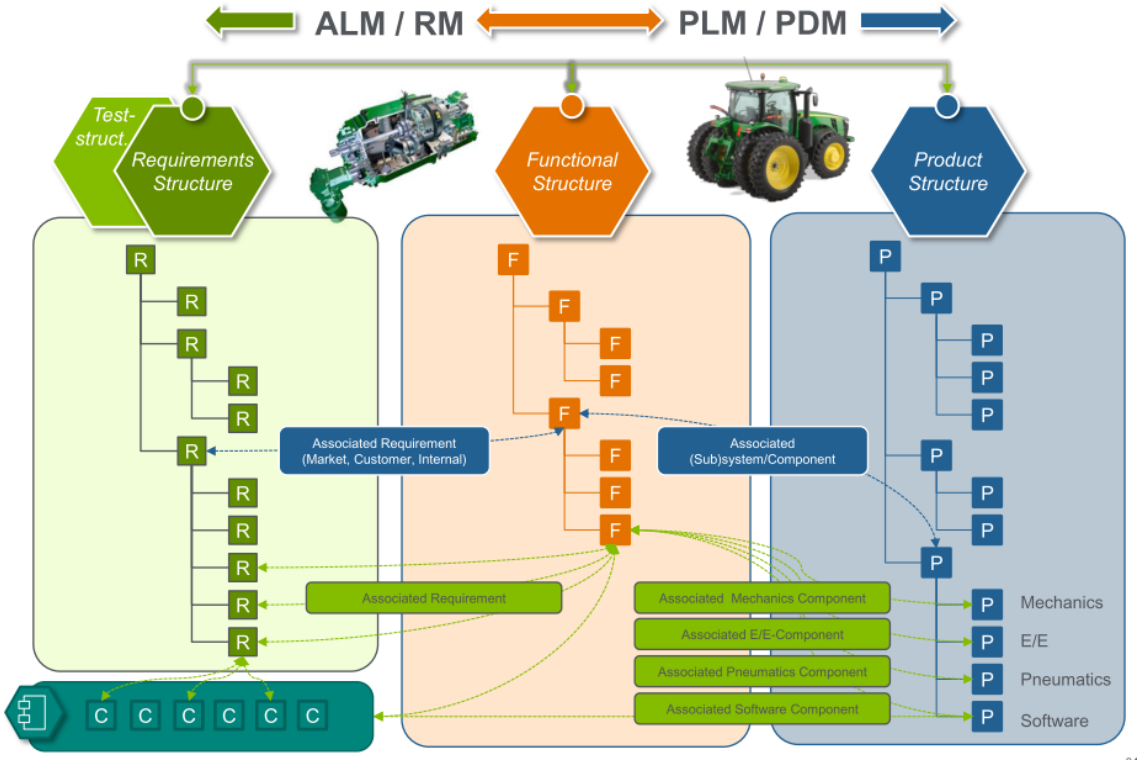

Figure 5-3: ALM/RM <=> PLM/PDM Mapping by PTC solutions[6]

[6] http://www.ptc.com/WCMS/files/158899/de/16_Systems_Engineering_Vision_-_Derek_Piette_Colin_White_PTC.pdf

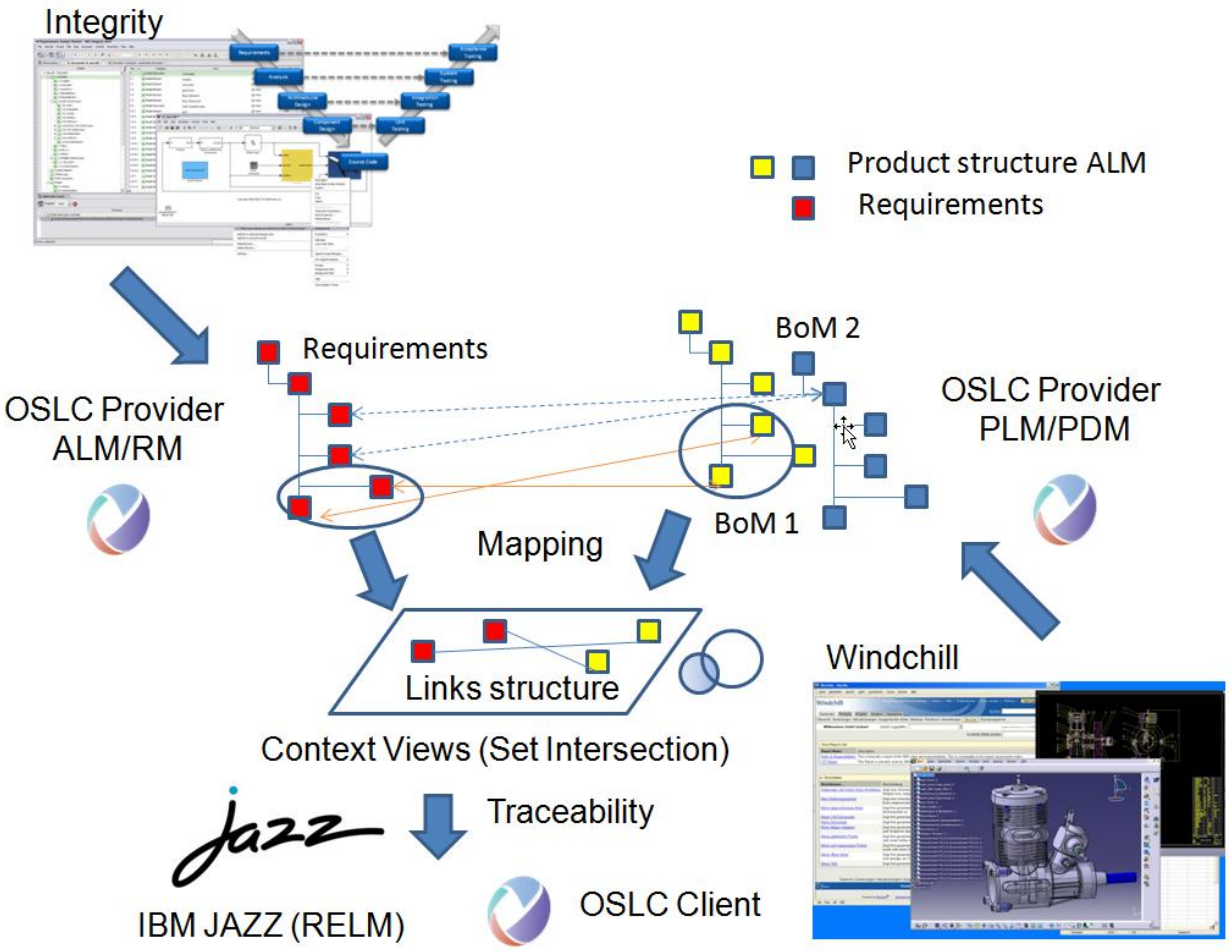

Figure 5-4: OSLC Solution concept for requirements and product structure mapping

A third possible solution combines both approaches: ontological linking and tool chaining based on OSLC Technology.

5.2.3.1 OSLC[7],[8]

Open Services for Lifecycle Collaboration (OSLC) is an open community creating specifications for integrating tools. These specifications allow conforming independent software and product lifecycle tools to integrate their data and workflows to support end-to-end lifecycle processes. The goal of OSLC is to create specifications for interactions between tools. Currently, specifications for ALM/RM (Application Lifecycle Management/ Requirements Management), CM (Configuration Management), QM (Quality Management), and several others are supported. OSLC is based on the W3C Linked Data[9] and offers two primary techniques for integrating tools – “Linking data via HTTP” and “Linking Data via HTML User Interface”. Both of these techniques build on the HTTP

[7] http://open-services.net/

[8] http://open-services.net/resources/tutorials/oslc-primer/what-is-oslc/

[9] http://linkeddata.org/

and RDF foundation of OSLC. As underlying base data Model OSLC uses RDF (Resource Description Framework)[10] as data model and as base infrastructure for services REST and HTTP.

5.2.3.2 Scenario solution based upon OSLC

As shown in Figure 5-4, an OSLC-based solution could include two OSLC providers and either one OSLC client that consumes the mappings made by the providers or a centralized solution using the IBM Jazz platform as central repository, where an infrastructure for searching, versioning, and visualizing of interlinked data is provided.

Regarding operability and the idea of an IOS Platform intended to by standardized within CRYSTAL project the third solution with OSLC seems to be most flexible because it combines the advantages of both previous two approaches: flexibility regarding, mapping and connecting structures as well creating views, as well as standardized service infrastructure to couple tools and exchange data between them.

5.3 Workflow oriented V&V - WEFACT

5.3.1 General Description

WEFACT is a workflow-oriented tool controlling validation, verification, and certification processes of critical systems and components as a basis for automated safety cases. Using IBM Rational Doors requirements management, WEFACT connects V&V tools and artifacts to be validated or certified with flexible instantiations of generic validation plans for the individual standards and safety integrity levels (in CRYSTAL for EN 50128 and EN 50129).

The “Workflow Engine for Analysis, Certification and Test” (WEFACT) has the goal to facilitate validation, verification, and certification of safety-critical systems in a modular manner.

WEFACT consists of the WEFACT framework which provides a flexible infrastructure for defining and executing the V&V process and the external resources – external processes, tools, and standards – which are integrated into the WEFACT framework by well-defined interfaces. Additionally, an extensive on-line user guide (“help file”) including a validation-plan (v-plan) cook book (“How to develop a v-plan”) is available.

The overall organization of the WEFACT framework is shown in Figure 5-4. The gray boxes show the elements of the WEFACT framework, the white boxes show the rest of the elements of the WEFACT (belonging to the external systems), vertical alignments indicate ‘uses’ or ‘consists of’ relationships whereas the arrows indicate major information flows.

The safety case is an argumentation to convince a licensing authority that a product is “sufficiently safe”. Typically, a safety case comprises the necessary safety arguments which correspond to the v–plans for each artifact under test (AUT) and the related evidence.

[10] http://www.w3.org/RDF/

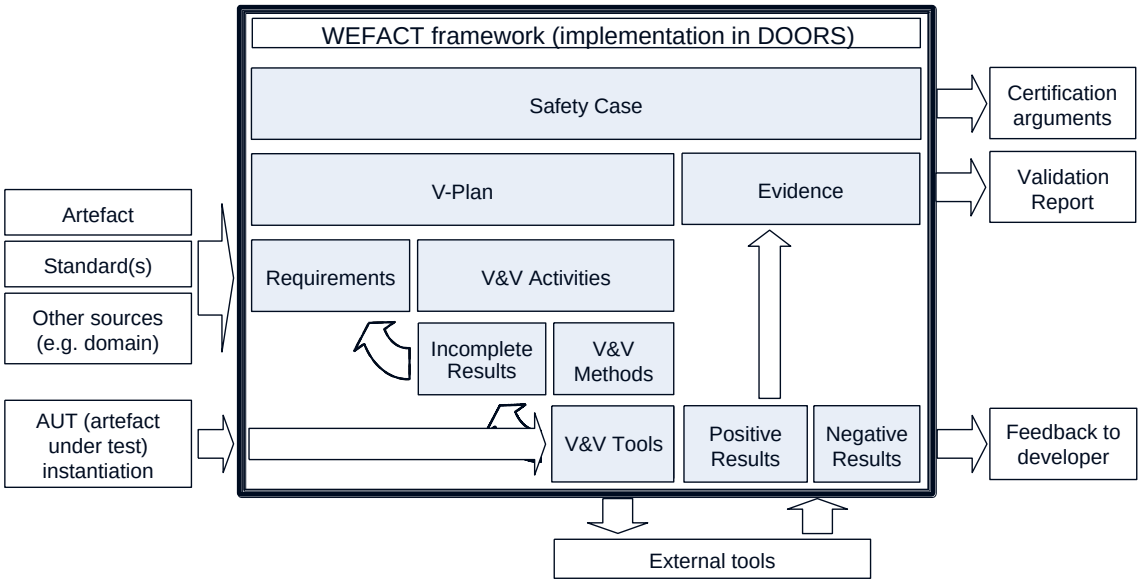

Figure 5-4: Overall organization of the WEFACT framework

The WEFACT framework is implemented with IBM Rational DOORS which is based on a distributed client/server architecture. Data such as v-plans, V&V activities, requirements is stored in the central DOORS database whereas the documents such as evidence and reports are stored in a separate document repository. In order to setup the WEFACT framework for a user, he or she installs the DOORS client in order to have access to the data and sets up the access to the document repository.

5.3.2 Data Flow

The WEFACT tool does not exchange data with other tools but rather manages links and references to elements of other documents, data base records, model elements, etc.

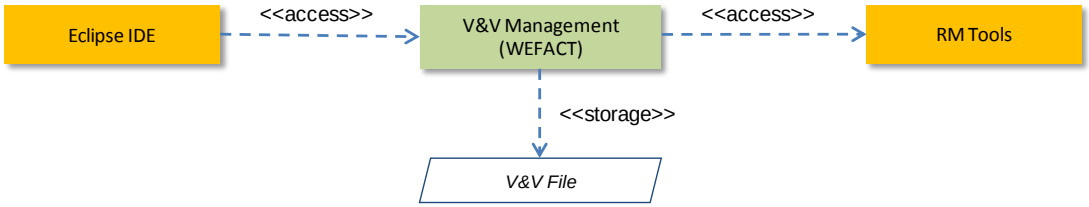

Figure 5-5: Dependencies of V&V management on other artefacts

Based on the CRYSTAL IOS, the tool will interface with any other requirements management tool (not only DOORS). Access from Eclipse tools to the V&V management will also be possible.

5.3.3 Integration into Use Case

See the deliverable D604.011 for the requirements of this use case. WEFACT will be further developed according to these requirements.

5.4 C²FT

5.4.1 General Description

C²FT stands for Component-integrated Component Fault Trees and is the evolution of Fault Tree Analysis (FTA) and Component Fault Trees (CFT). This technique has been created with the aim of facilitating fault tree analysis during the design process. This is achieved by the C²FT approach by defining a formal relation between a CFT and a component in a component model. This relation is not only established between the two models but also between their interfaces, so that failure modes of the CFT are associated with the incoming and outgoing interfaces of components. See Figure 5-6.

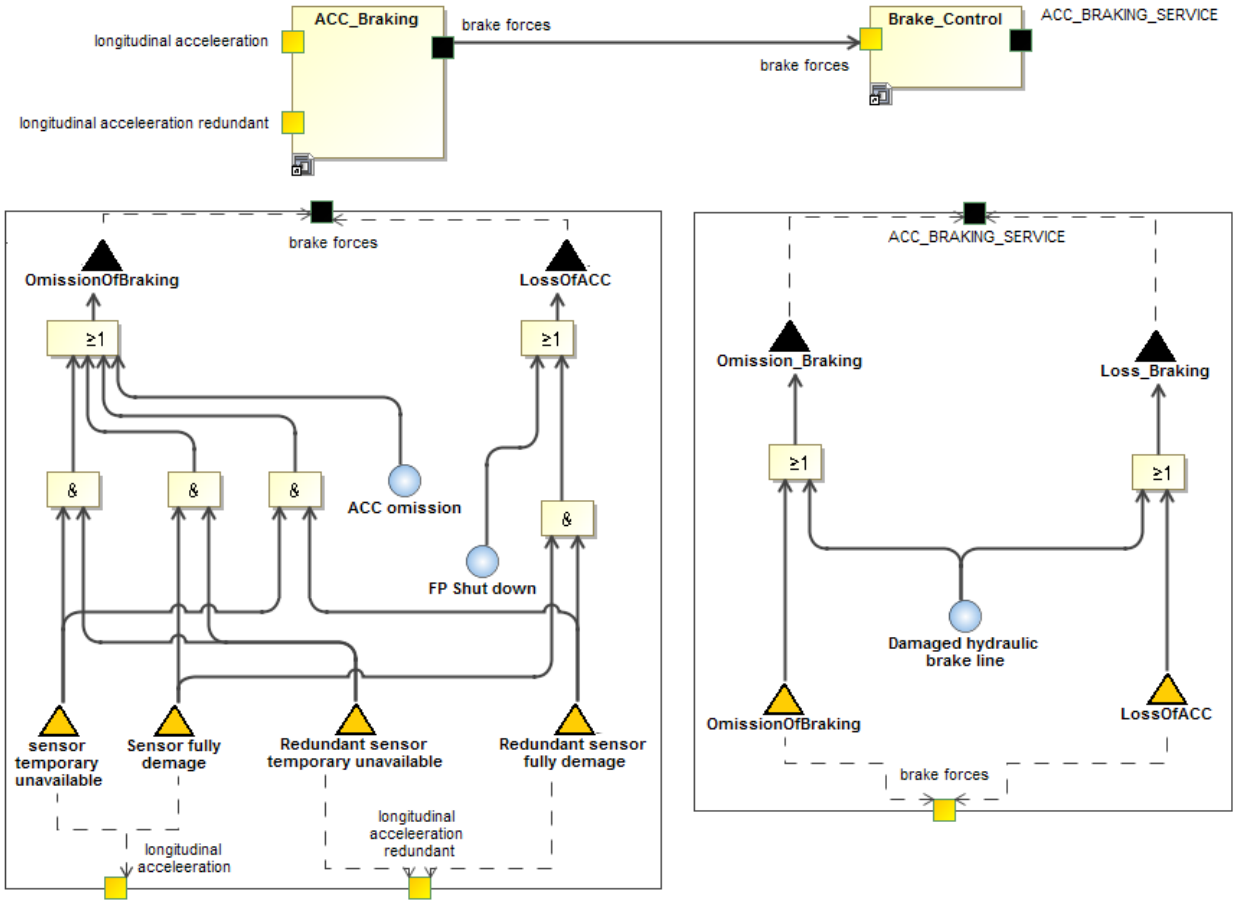

Figure 5-6: C²FT example

The formalization of the relation between the safety model and the component model, represents and enhancement with respect to the predecessor techniques regarding:

- Consistency
- Traceability
- Maintainability

- Reusability

Furthermore, C²FTs also help handling the complexity of the safety analysis by keeping the same modular and hierarchical structures as can be defined in the component/architectural models.

An implementation of the technique already exist in form of plug ins provided as an extension of the commercial tool Magic Draw[11].

5.4.2 Integration into Use Case

In CRYSTAL C²FT will be extended and integrated into the IOS/RTP. Concrete extensions and integration scenarios can be found in deliverable D604.011.

The concrete task we fulfill in this Use Case is that we support functional safety analysis by applying C²FTs as functional safety analysis technique. By this we complement the afore mentioned FMEA analysis and on the other hand show the potentials and benefits one gets when using safety analysis models that are tightly integrated into system models.

5.5 Knowledge and Information database

Due to the complexity of the tooling and the many variants that the projects span, with a mixture of common and variable information, Infineon are investigating the best manner of controlling and ensuring correctness of the data within the projects.

The main data sets have been gathered into an excel spreadsheet to begin with, while the type of data is then analyzed for where and how it is needed and will be controlled.

The aim of KID is for central control of this data, with a simple interface which can ensure up-to-date and correct information, from where all tooling can gather information.

5.5.1 Current Requirements Analysis for KID

The current analysis of information that may be incorporated or that is a mandatory field of information to ensure tool integrity:

- Hierarchical data
  - RequisitePro -> Safety Concept -> Internal target specification -> Testplan -> results, knowing which section relates to which within the documentation tree is required for tooling such as DaD B3.91a (the data is essentially a Lookup table for this tool)
- Path data
  - Information on where data resides within the configuration management systems.
- Extraction Commands

[11] Magic Draw is a software and system modeling tool developed and distributed by No Magic Inc.

    - What call is required to extract data from different databases. Currently these are similar but have small id name differences which may change dependent on the database from where the data comes
- Roles and responsibilities
    - Management, Concept engineers, Verification engineers responsible on a per project and per system, sub-system, module instance.

Much of this information is already contained and managed within separate well-established databases and as such this should not be replicated as any manual replication of data will lead to unintended data integrity issues which would cause tooling issues.

- KID shall enable a central information source which shall allow easy interfacing to and for all necessary tools
- KID shall have access to all relevant current databases
- The linked databases will be interfaced by their reference so that any name changes etc will be incorporated without an update to the database itself.
- Interfacing to many of the Requirements engineering tools within IFX-UK is done via ARQE.xml so the IOS liking the databases must support this schema
- KID shall be strictly controlled via an interface to a change control system (in IFX this will be Jira).

- Interfacing between the databases may make use of OSLC
- Restricted access for admin and less restricted on a per role basis shall be enabled.
- Long term database management shall be planned
- A central Eclipse based dashboard is planned for access to all project databases – KID shall be included within this project
- Configuration and historical management shall be enabled
- KID shall allow the production of quality documents for audit purposes.

5.5.2 Integration into the use case

See chapter 6.4.3.1 in the automotive public use-case figure 3.5, which shows KID and its interoperability within the IFX-UK Data Flow Diagram

6 Interoperability Requirements

Based on the Use Case description we identified several general interoperability needs / requirements which might be relevant to an interoperability specification (IOS). The aim is that the IOS and its concepts eases the instantiation of the Reference Technology Platform (RTP) that fulfils the UC needs and in particular the interoperability aspects of the UC.

The UC is focusing on systems engineering within the V-Model in particular requirements specification and V&V methods. Several tools will be used within the tool chain of the UC. In terms of interoperability and integration aspects the solution will be based on the PTC Integrity framework. The focus of the UC is also shown in the following figure.

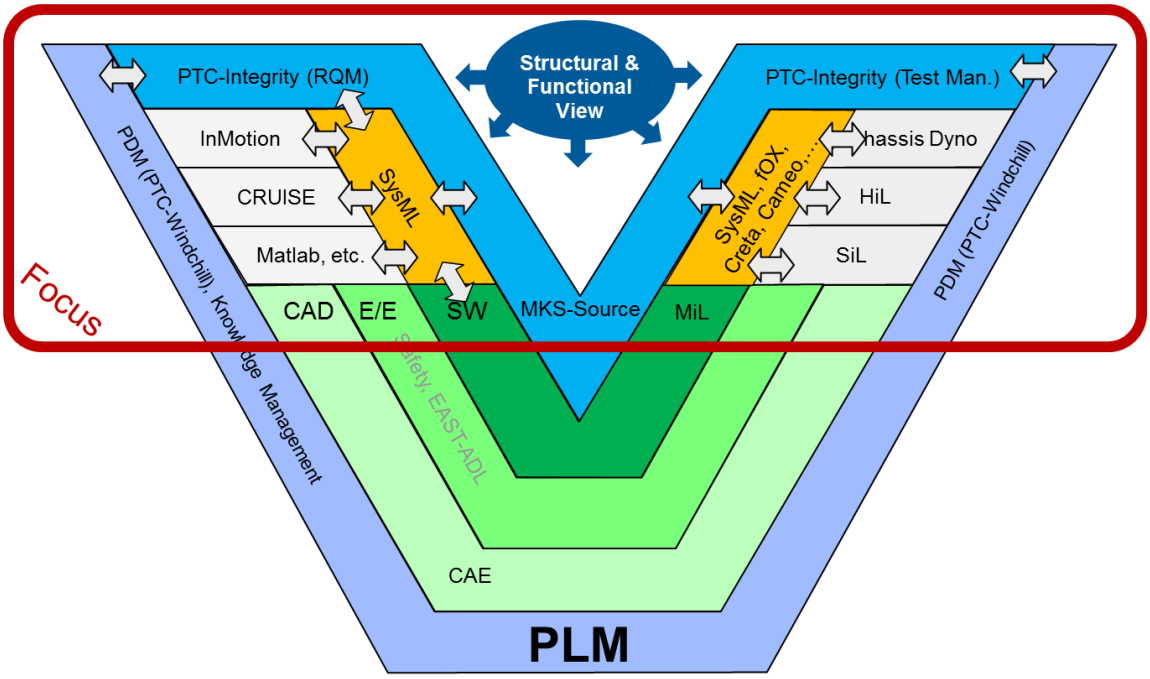

Figure 7: Framework & Interfaces for Systems Engineering Tool-Chain & Interoperability Requirements

In case of using PTC Integrity and SysML several tool connections or interfaces are needed. These interfaces are described in the following section.

6.1 General UC Interoperability Needs / Requirements

The following section summaries the main interoperability requirements which are relevant to the UC. The aim is that the IOS and RTP solution covers these requirements on a certain level in order to ease the realization of the UC based on the CRYSTAL approach.

6.1.1 Traceability

The linking of all engineering artifacts is one of the major interoperability requirements of the UC. We identified several aspects which shall be addressed by the IOS Traceability approach.

Therefore, the CRYSTAL IOS shall support traceability. The traceability solution shall allow creating typed links between elements. Traces shall be bi-directional and navigable in order to apply appropriate analysis methods, such as requirement coverage. Furthermore, the traceability information shall be consistent and available at all phase of the development process. It shall be possible to import legacy traceability information to the new CRYSTAL approach.

6.1.2 Tool and Data Interoperability

Huge amounts of information data will be produced during the development process by a lot of heterogeneous engineering tools. These tools shall exchange the produced information seamlessly. Exchanged information shall be manageable in a consistent way within the different tools. The UC requires establishing interoperability between the following tools: PTC Integrity, AVL InMotion, AVL Cruise, and Matlab Simulink. The UC will mainly share artifacts such as requirements and tests.

6.1.3 Versioning Support

The huge amount of produced information data within the UC demands a proper mechanism to handle and to manage these data. A suitable solution is to versioning all development artifacts. This includes the synchronization and merging of diverse development branches. Therefore, the UC is requiring an IOS approach that supports version management of development artifacts produced and exchanged by different tools within the UC tool-chain.

6.1.4 Multi-User support and User Collaboration

One major challenge in the powertrain engineering process is that many users need to collaborate across locations and engineering disciplines. Therefore, the UC requires an IOS and RTP solution which supports multi-users in distributed environments. Hence, the UC requires a mechanism to synchronize users and to ease the user collaboration.

6.1.5 Distributed Development

Obviously, the UC tool chain is a distributed system with typical challenges of distributed systems. However, the UC requires an IOS approach that supports the maintainability, consistency as well as availability of necessary engineering information.

6.1.6 Automation Support

Within the UC, several engineering methods are applied which could potentially be automated, such as requirements coverage analysis. Therefore, the UC requires an IOS approach that supports the automation of engineering methods as services. These services shall be easily integrated into the engineering process.

6.1.7 Analysis Support

The UC makes use of a high number of engineering methods dealing with analysis, such as hazard and risk analysis or failure mode and effects analysis. Hence, the IOS and RTP approach shall support these analyses by providing appropriate methods in order to automate, if possible, and to configure the analysis in a useful way. For instance, if specific input data for an analysis is requested, that data shall be transported transparently from the user to the tools which will perform the analysis.

6.1.8 Simulation Support

Simulation is also an important engineering method to get assistance in several dimensions like calibration or testing in order to reduce the development time and costs. Therefore, the UC is requesting support for simulation activities within the IOS principle. Simulation parameter and results shall be shared within the UC tool-chain.

The following sections describe more detailed needs towards tool interoperability.

6.2 SysML – AVL CRUISE Interoperability Requirements

An interface between a SysML tool and AVL CRUISE allows the detailed simulation of powertrain systems in AVL CRUISE which is based on a central system model established in SysML. This guarantees the usage of an established system model without loss of information during simulation. Results of sophisticated investigations of simulation engineers are mapped back to the system model. Interactions and consequences on system, components, and interfaces are highlighted within the SysML environment without adding complexity to the system model.

In this context the following principle roles are assigned to the SysML tool and AVL CRUISE:

- SysML works as donor of the original system model. The required simulation view is fully integrated into the cross-discipline system context, interfaces. Base model and parameters originate from one common source of information
- AVL CRUISE maps the original system model. CRUISE refines and executes the system model in the simulation environment. Required simulation results are transferred back to the SysML tool

To facilitate the seamless information exchange described above, a bi-directional interface between both tools shall fulfill the following interoperability requirements:

- The required base information for simulation activities shall be applicable from the SysML model (e.g. powertrain configuration, target values)
- The SysML objects shall be automatically mapped on CRUISE objects including base parameters, object interfaces, and connections
- Targets (parameters, figures,…) shall be selectable within the SysML tool to define the required outcome from the simulation study
- Changes in the simulation model which affect the original SysML model shall be highlighted in the SysML environment to allow an investigation how the changes affect the system
- Simulation-specific information which is needed to run the system simulation in the CRUISE environment shall not be mapped to the SysML in order to reduce the complexity of the system model
- The target objects shall be transferred back to the SysML model
- The interface shall be configurable (synchronization procedure, quantity of input objects and results) to allow project dependent applications

6.3 Integrity Interoperability Requirements

6.3.1 Needed Interfaces

Interface	Artefacts
Integrity – Artisan	Bi-directional links/Traces Synchronization of Model Elements / SysML Tool-spanning Change Management Configuration Management of Artisan Models in Integrity
Integrity – Windchill	Bi-directional links/Traces Synchronization of System Requirements Tool-spanning Change Management Tool-spanning Test Management
Integrity – Testing Tools	Bi-directional links/Traces Import of Test Results into Integrity Defect Management: Creation of Defects in Integrity triggered from Testing Tools

6.3.2 Interoperability Use Cases

6.3.2.1 Integrity – Artisan

Considering the engineering methods described for WP303, the following interoperability use cases are to be expected for the Integrity - Artisan interface:

It should be possible to manage versioned Artisan models in Integrity. The interface should allow several users to work in parallel in their SysML model sandboxes. Once a user has finished his work he must be able to check-in his changes (full model or only a part of it) and the interface takes care of the correct configuration management in Integrity.

The interface should allow a representation of a SysML model package hierarchy in Integrity that contains just enough information to allow the creation of traces / bi-directional links between Artisan model elements and other artifacts of the lifecycle (requirements, test cases, etc).

The interface should allow tool-spanning change management and impact analysis. This means that if something changes, for example a requirement is modified, the Artisan user can find out by applying an impact analysis which parts of the SysML model needs to be changed.

6.3.2.2 **Integrity – Windchill**

Considering the engineering methods described for WP303 the following interoperability use cases are to be expected for the Integrity - Windchill interface:

It should be possible to exchange system requirements (both single requirements and complete documents) between Integrity and Windchill. Links/Traces between requirements should also be synchronized.

It should be possible to create bi-directional external links between artifacts in Integrity (i.e. Test Cases) and Windchill artifacts (i.e. BOM elements).

The interface should allow tool-spanning change management and impact analysis. This means that when a change is introduced in e.g. a requirement the Windchill user can find out via an impact analysis which parts of the BOM model he has to.

The interface should allow tool-spanning test management such that it is possible to plan and develop tests spanning both artifacts managed in Integrity and Windchill. It should also be possible to execute those tests and manage test results for them.

Minimal requirements for these tasks would require uniform standardized interfaces (API Form) for the tools as well as wide-spread schemas (e.g. Ontologies, OWL, RDFS) and data models (e.g. XML, RDF) for mapping to functionality structure. As interaction layer for data exchange services running on HTTP and REST in order to make the exposure of the data also to other participants on demand possible.

6.3.2.3 Integrity – Testing Tools

Considering the engineering methods described for WP303 the following interoperability use cases are to be expected for an interface connecting Integrity and external testing tools:

Test cases, test results, and calibration data should be exchanged between Integrity and the following tools: fOX, Cameo[12], Creta[13]. In case a test case fails, the interface should allow to automatically creating defects in Integrity.

[12] See also WP6.3

[13] See also WP6.10

7 References

[Eigner et al. 2005] M. Eigner, R. Weidlich, M. Zagel. (2005) The Conceptual Product Structure as Backbone of the Early Product Development Process, Proceedings of the ProStep iViP Science

[Haberfellner et al. 2002] Haberfellner, Nagel, Becker, Büchel, von Massow, Hrsg.: Daenzer, W. F.; Huber F.: (2002) System Engineering – Methodik und Praxis, Verlag Industrielle Organisation Zürich, 11. Auflage.

[Pahl et al. 2007] G. Pahl, W. Beitz, J. Feldhusen K.H. Grote (2007) Engineering Design. Springer Verlag London, ISBN-10:1846283183

[Rosenberger & Denger 2009] M. Rosenberger, A. Denger (2009) Semantic Structure Mapping in the Earlier Phases of the Product Lifecycle, Proceedings of the International Conference on Product Lifecycle Management, PLM'09, Supporting the extended enterprise, PLM-SP5, 2009 pp. 598 – 608.

[Schuh 2005] G. Schuh. (2005) Produktkomplexität managen. Hanser Verlag, 2. Auflage

[Hitzler et al. 2008] P. Hitzler, M. Krötzsch, S. Rudolph, Y. Sure. (2008) Semantic Web. Springer-Verlag Berlin Heidelberg.