Contract number: ITEA2 – 10039

Safe Automotive soFtware architEcture (SAFE)

ITEA Roadmap application domains:

Major: Services, Systems & Software Creation

Minor: Society

ITEA Roadmap technology categories:

Major: Systems Engineering & Software Engineering

Minor 1: Engineering Process Support

WP 6, WT 6.1.2 and 6.2

Deliverable D.6.a

Methodology and application rules Documentation

Initial Documentation

Due date of deliverable: 31/12/2013

Actual submission date: 28/02/2014

Start date of the project: 01/07/2011 **Duration:** 36 months

Project coordinator name: Dr. Stefan Voget

Organization name of lead contractor for this deliverable: Continental

Editor: Jörg Kemmerich

Contributors: Philippe Cuenot, Dr. Thomas Peikenkamp, Dr. Thomas Wenzel, Maged Khalil, Dr. Alexander Rudolph, Dr. Juergen Lucas, Jörg Kemmerich, Dr. Stefan Voget, Hans-Leo Ross, Andreas Eckel; Elvira Biendl, Nico Adler, Stefan Otten

Reviewer: all safe member are invited.

Version	Date	Reason
0.1	2013-01-29	H.L. Ross - Initialization of document
0.2	2013-02-04	S. Voget - Review regarding project task objectives
0.3	2013-02-06	H.L. Ross – Introduction of safety mechanism in 5.1.7
0.4	2013-04-03	H.L. Ross – Draft Info in German.
0.5	2013-04-19	H.L. Ross – Modified Draft Info in German
0.6	2013-06-13	J. Kemmerich, K.Niewiadomski – Integration of review inputs
0.7	2013-06-13	Complete review of deliverable D6.2
0.8	2013-07-25	S. Voget - Complete restructuring after workshop
0.9	2013-10-03	Ph. Cuenot – Contribution chap 4.3.3, 5.1.2 and 5.1.7
1.0	2013-10-31	K. Niewiadomski: Contributions by AVL, BMW Car-IT & ZF added
1.1	2013-12-16	K. Niewiadomski: Contribution by Conti Teves (chap 10.2.7) added
1.2	2014-01-06	A. Rudolph: Contributions on industrial standards (section 4.1) and “malfunction/loss of function” (section 5.1.6)
1.4	2014-02-26	K. Niewiadomski/J. Kemmerich: Integration of content and finalization
1.5	2014-02-28	J. Kemmerich: reworked version for convertation into PDF document

1 Table of contents

2 Executive Summary

The objective of this work package is to tackle the introduction of an information flow combining the work products requested in ISO26262 to a real engineering team. Based on this information flow, an assessment methodology for functional safety is specified, which accompanies the development process until safety validation, also taking into account the collaboration of OEMs and a tier one suppliers or tier 1 and tier 2 suppliers. Work-products and safety activities realized by the SAFE project and adequate measures are documented to allow seamless implementation in the different engineering disciplines. This information flow is evaluated during use case evaluation or other available use cases according to the above objectives.

The model based technology is introduced in a second step to perform adequate engineering steps and verifications required by the assessment measures, in order to benefit from developed techniques and accelerate development process steps to satisfy standard requirements.

The second activity of this work package is to make available a series of guidelines for the use of the methods and tools developed in the SAFE project. Starting from the analysis of the different industrial development scenarios, an exhaustive list of recommendations and guidelines is provided for the development of a safe automotive architecture. These application rules detail best practice, standard patterns, and concrete example to document specific highlight of the safety standard applied in context of product development.

More specifically, the application rules address the following topics:

- Decomposition recommendations for effective design of safety mechanisms
- Compliance with architecture constraints and safety mechanisms and supervisor architectures
- AUTOSAR platform configuration for safety
- Inclusion of COTS in a system developed according to the ISO26262 standard
- Application rules for mixed criticality approach.

In addition, application rules for the mixed criticality approach contain decomposition recommendations and instructions how to use and integrated the software layer into a system using AUTOSAR basic software components in combination with the safety layer.

This document will show how to proceed to satisfy overall ASIL-D requirements despite the use of non ASIL-D components (AUTOSAR basic software components) such system using the safety layer concept.

2.1 General description of assessment activity/architecture model for functional safety development (AAM)

Target is a reference process model for functional safety assessment activities based on required functional safety activities according to ISO 26262 and the description of the methodology. This goal shall be achieved by the delivery of an assessment activity/architecture model for functional safety development (AAM). The AAM provides a reference performing a assessment according to ISO26262. In particular the AAM consists of all safety activity and the data flow between them.

The methodology is based on results from the concepts and delivers templates and guidelines to apply automated model-based verifications (in the meaning of ISO 26262).

Based on analysis of the standard and required measures and considering the overall automotive supply chain, templates for verification planning are created. These templates show how the concepts support the safety activities mentioned in the verification plan.

This is done at all levels (incomplete list: HW component level, SW component level, system level), i.e. by defining the safety-related inputs/outputs that are required at each of the design stages.

Criteria and concrete measurements of a process (based on activities in the templates) are provided to verify e.g. the completeness of assessment.

The AAM is closely related to the result of the guideline and the collected methods linked in those guideline. The analysis of dependent failure is taken as an input for identification of the structure for AAM. The AAM provides at the end further content to the guideline, as can be seen in more detail in chapter 9.

2.2 General description of SAFE Engineering Process (SEP)

The SEP defines reasonable sequences of AAM that are derived from the methods (reference for the application of the ISO26262 standard).

For this a reference process for the model based development of safety relevant systems are identified. This reference process integrates and concatenates the methods and reflects the specific techniques developed in parallel in the first subtask.

Main references for this process are EAST/ADL and AUTOSAR meta-models and methodologies.

Results from the ATESST2 and EASIS project are taken into account in order to establish the reference process (SAFE Engineering Process, SEP). The parts of SEP are allocated to levels of the EAST-ADL and AUTOSAR meta-models and methodologies.

Process steps with referenced work products are documented. This reference process focuses on portions that are important for ISO26262. The outcome of this work package constitutes a reference for the application of the ISO26262 standard.

The process description starts with requirements engineering and ends with the start of production. The description should enable a process manager to provide a company specific process description to fulfill safety requirements.

The process is modelled using Enterprise Architect. These details are presented in the appendix.

3 Purpose and Scope of this document

This chapter provides a general overview of the SAFE Project, introduces the SAFE Meta Model and clarifies the purpose, boundaries and conditions applying to this document.

3.1 Fundamentals for this document

The SAFE project aims to address steadily increasing functional features and propulsion trends in current and future vehicles. This increase translates to complex software architectures, that require multiple views and abstraction layers to describe and necessitate complex analyses, mandated by safety standards, involving a large and occasionally disparate amount of information.

In order to carry out these analyses in a model-based approach, it is necessary to capture the required information in the Meta-model. Seeing as the analyses, and thus the information to be captured, range from abstract architecture descrtiption, through requirements refinement and on to hardware metrics and SW/HW components, it becomes necessary for the Meta-model to be able to capture all this information. The information necessary was collected by analyzing the ISO26262 safety standard, among other sources, and captured and refined in the form of requirements for the SAFE Meta-Model in WP2.

The requirements where then further refined and concentrated according to the project scope. SAFE focuses on facilitating the development of safe software architectures. As such, requirements pertating purely to process issues where excluded from the start, because they are not only not pertinent to the project, but also vary among different companies. Some further exclusions where made for scoping purposes and adjustments made due to project partner changes.

3.1.1 The SAFE Meta-Model

To not reinvent the wheel, existing modeling frameworks and architecture description languages (ADL) where analyzed for suitability to the required purposes. Chief among these, EAST-ADL, which is an ADL optimized for top-down description in the automotive domain and the subject of numerous previous as well as ongoing expansion and refinement research projects and described in more detail in Chapter 4.3.2, was found to cover many (but not all) of the aspects required, especially those pertaining to software architecture description.

It thus served as a basis for the SAFE Meta-Model, which is introduced as an extension package to EAST-ADL.

This only got us halfway, as EAST-ADL remains a fairly abstract description and all analyses and information have to be assigned to components at the end of the line. In order to describe SW/HW components we selected AUTOSAR. AUTOSAR is an open and standardized automotive software architecture, jointly developed by automobile manufacturers, suppliers and tool developers, to facilitate and standardize software communication, transfer and maintainability across hardware platforms. It is a bottoms-up approach, described in more detail in Chapter 4.3.1.

IP-XACT and similar hardware descriptions where analyzed for hardware analysis information.

To undersand the construction of the SAFE-Metamodel several points must be understood:

1- It is necessary to understand that none of these modeling languages and, more importantly no even the sum of them, completely cover all the SAFE requirements.
2- EAST-ADL does not map directly onto AUTOSAR. There are large gaps in many areas, much (occassionally conflicting) overlap in some areas, and some areas that do not map. Essentially, EAST-ADL and AUTOSAR are orthogonally transposed. Mapping EAST-ADL to AUTOSAR is thus best done within a specific context. In this case the context is functional safety.

As such this gives rise to three kinds of artifacts in the SAFE Meta-model:

1- 1:1 mapping onto existing EAST-ADL and AUTOSAR artifacts; the exisiting artifacts perfectly suit the identified needs.
2- Extensions of pre-exisiting artifacts; the exisiting aritfacts provide a solid basis but some necessary information is missing.
3- New Artifacts; this necessary information was not available or adequately covered in the existing modeling languages.

The last two types of artifacts have led to highly constructive discussions with relevant standards commissions and most have been transformed into change requests as precursor of inclusion into the relevant standard.

3.1.2 Use of the SAFE Meta-Model

As previously explained, the SAFE Meta-model does not explicitly address matters purely pertaining to process isuues (e.g., Change & Configuration Management..) and while it is highly comprehensive, it does not cover ALL the activities or produce all the artifacts prescribed in the ISO26262 Safety standard. Thus, the purpose of this document is:

1- To define application guidelines, which explain how the Meta-model methods defined and specified in WP3, and integrated and operationalized in WP4 are intended to be used.
2- To define an engineering process model, which explains how the numerous activities, and the corresponding generated artifacts, covered by SAFE can be integrated into a generic process (SAFE Engineering Process SEP), independent of organizational structure.
3- To define how the SAFE Meta-model and its generic SAFE Engineering Process SEP can be employed for a safety evaluation.

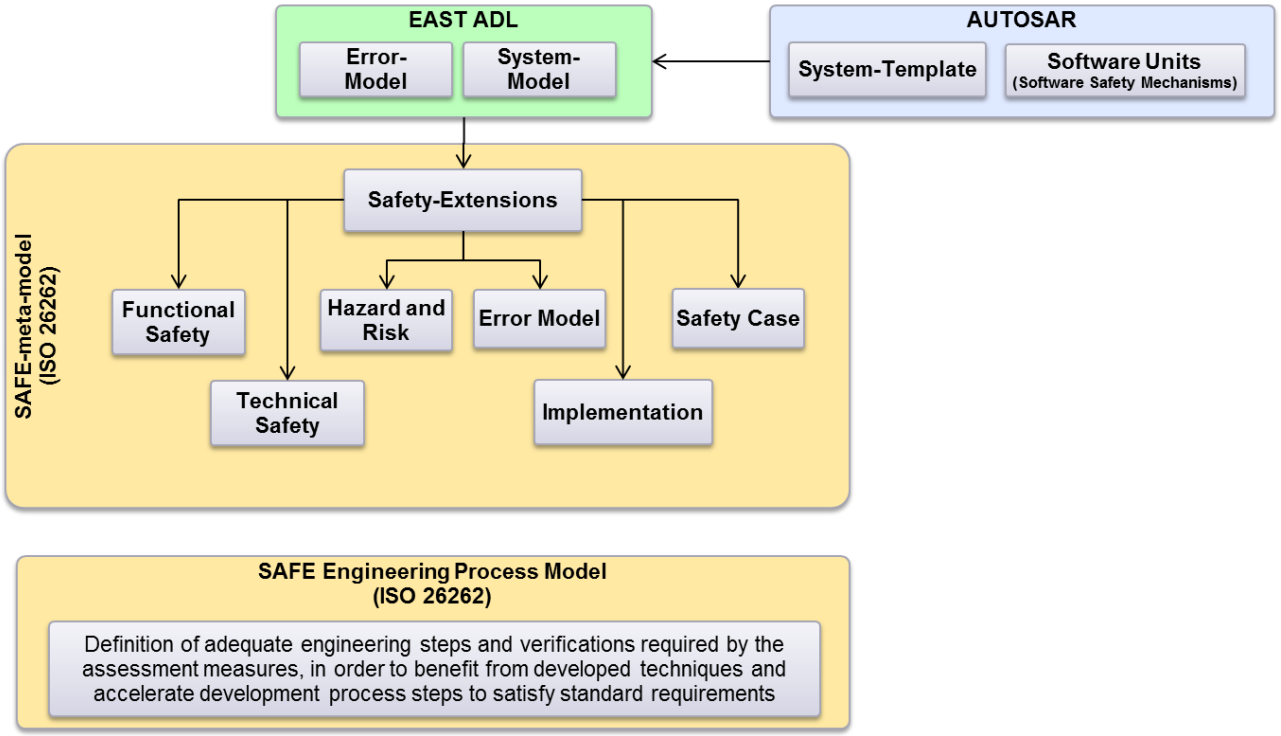

Figure 1: SAFE Meta-Model Extension

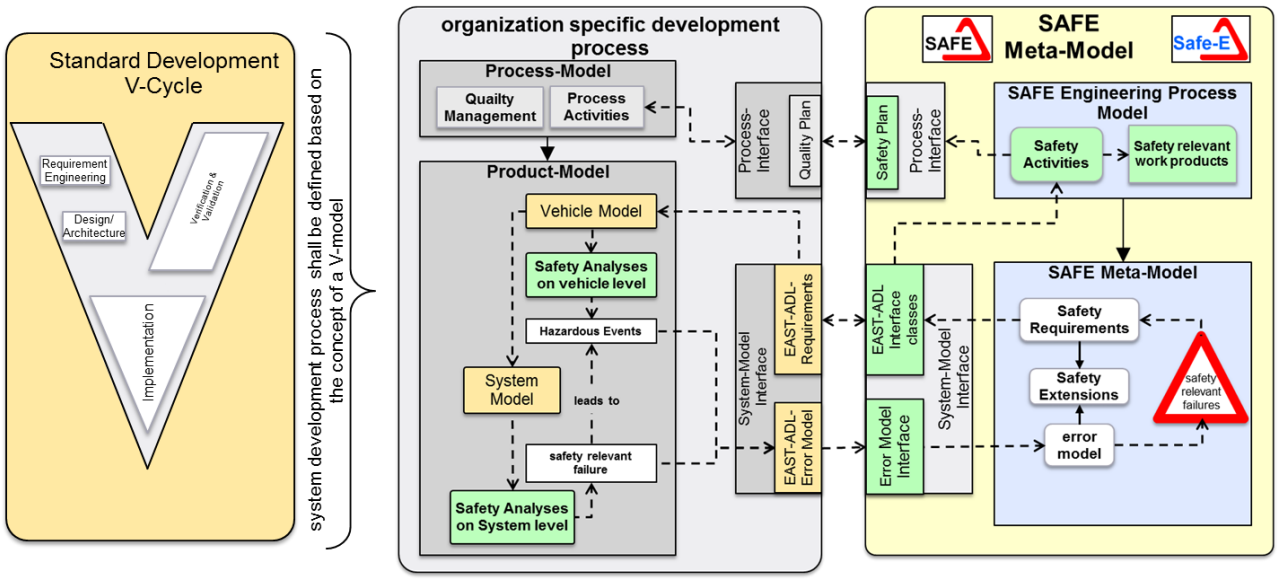

Figure 2: SAFE Meta-Model Use Case

3.2 Organization of document

The document starts with introduction and collects needed technologies to describe the guideline. The core of the guideline is oriented on the structure of the safety life-cycle defined in the ISO26262.

Each activity is described by using an unique template with the following structure within Chapter "C":

C.1 Guidelines and activities in a ISO 26262 Phase

In this subchapter those activities relevant to the ISO26262 Concept Phase, which can be carried out using the SAFE Meta-model and the SEP, are explored in detail. Usage guidelines and limitations as well as exemplary implementations are provided.

Initially a concise explanation is given of what the ISO definition of this step is.

C.1.1 Item Definition
Short explanation of what the step means according to ISO26262.

C.1.1.1 Activities (Relevant activities using the SAFE Meta-model)

Explain concisely what activities the ISO requires and then state which of them are supported by the relevant SAFE Meta-model parts.

C.1.1.2 SAFE Meta-model Formalism

Explain the meta-model at modelling level, including artifact interfaces etc.

C.1.1.3 SEP Formalism

Explain the mapping of the activities onto the generic process model.

In addition to that the following two chapters can be added:

C.1.1.4 Exemplary tool usage

Provides, where possible, tool implementation examples provided by the SAFE partners, which showcase the operationalization of the method supported by the meta-model.

C.1.1.5 Exemplary industrial use case

Provides, where available, examples from the industrial use cases carried out in the SAFE project, which were able to (successfully) employ the described methods/activities.

3.3 Architectural structure principal

The SAFE Meta-model was created based on the architectural structure principals given in EAST-ADL and AUTOSAR that are used in automotive industry.

3.4 Relation between Process, Methods, Tools, Environment and People considered.

The figure 2-1 from "**Survey of Model-Based Systems Engineering (MBSE) Methodologies**" from "**INCOSE MBSE Focus Group"** provides a basic idea how the relation between process, methods, tools and their environment of use could be considered.

The following figure is an overview of relations between user of technologies using questions to categorize them. Not all interfaces are considered within the project. Aspects concerning people and environment are not considered.

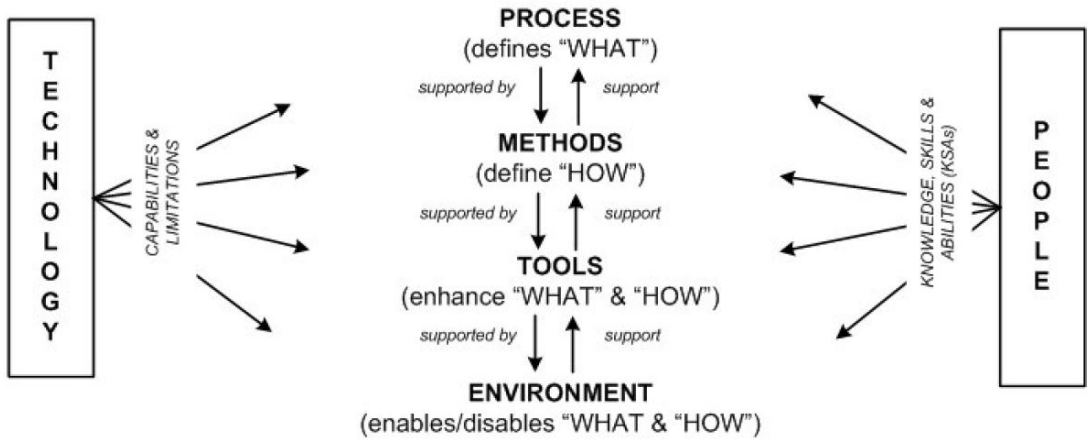

Figure 2-1. The PMTE Elements and Effects of Technology and People.

Figure 3: PMTE Elements and Effects of Technology and People

4 State of the art

Analysis is performed to take results of other projects into account and develop/maintain a kind of general process for product development.

4.1 Standards

Technical standards compiled and authorized by international organizations are the vital framework of safety-critical or safety-related system/product design and operation in all industries. As summarized in Figure 4 below, a central role is assumed by IEC 61508 in almost all industries, also a legitimate ancestor of the ISO26262 derivate in the automotive domain.

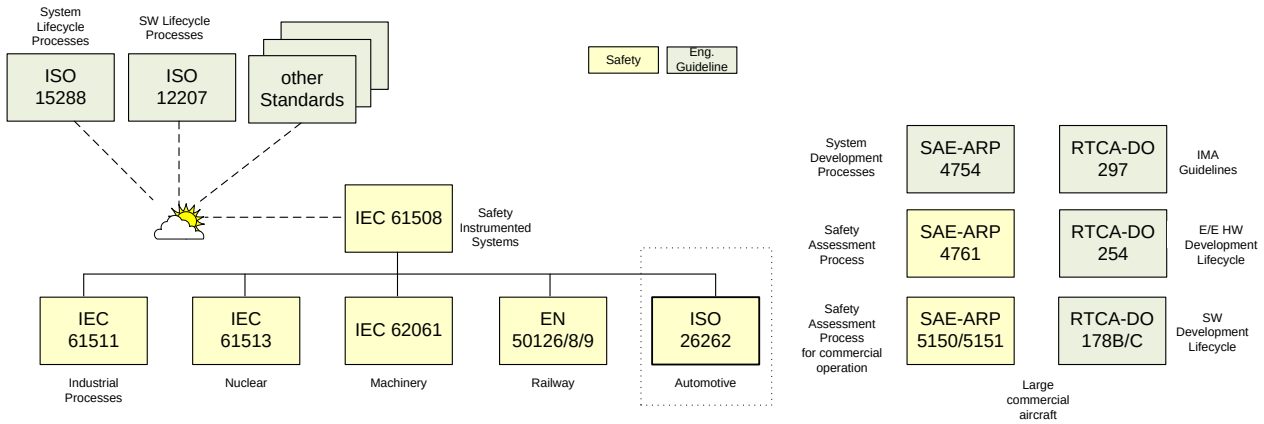

Figure 4: Technical Standards of dedicated Industries

Nevertheless ISO26262 attempts to integrate aspects from the unified framework of the large commercial aircraft domain[1]. One might consider this problematic, as the more or less structured systems engineering processes are not available outside the airborne world.

Well-known standards as e.g. RTCA-DO178 do not strictly focus on Safety itself but on the development of items (here computer SW) classified as safety-critical by top-down system design processes governed by SAE-ARP4754A.

Basically available standards as ISO15288 are in no way integrated in the overall engineering process, frequently leading to a condensation of “safety design” on SW level. Due to the rapid increase of SW intensity, a common error observed these days is the assumption that Systems Engineering is covered by Software Engineering. In terms of Safety this misconception is at worst adverse, as fulfilling SW standards does in no way mean designing a safe embedded system.

So, ISO26262 tries to bridge a large gap here; as an equivalent to the airborne SAE-ARP4754A systems engineering framework is widely missing.

[1] Which is regulated by the CS25 certification specifications issued by the JAA/EASA airworthiness authorities. Their paragraph CS25.1309 details the scope and content of the safety analyses to be conducted. So in commercial air transport, certification includes safety. This is different to e.g. automotive, where the ECEs applicable for homologization do not contain any obligatory link to ISO26262.

4.1.1 ISO 26262

The ISO 26262 defines rules and processes that are used as a base to derive this guideline. Reference for all activities related to ISO 26262 is the safety lifecycle from ISO 26262, Part2, figure 2.

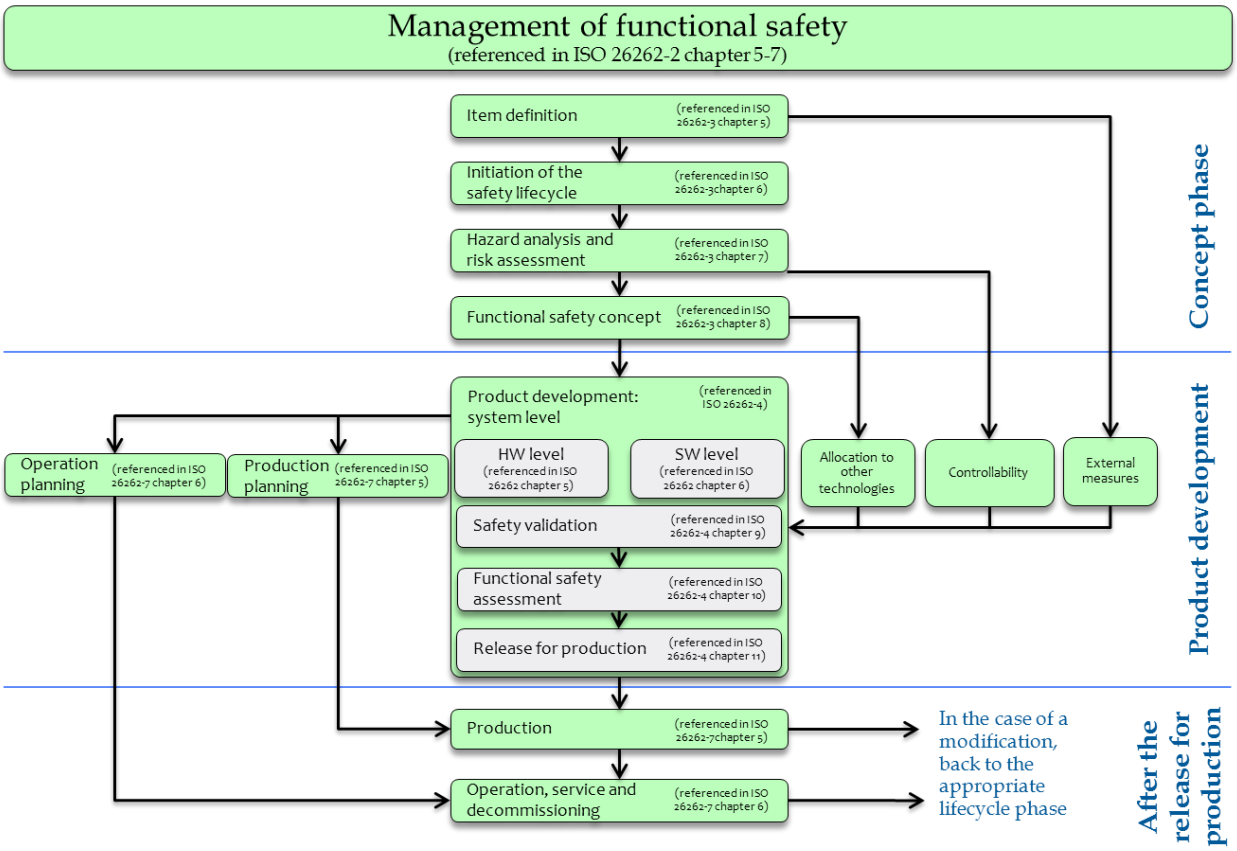

Figure 5: Safety Lifecycle (screenshot out of ISO 26262)

The safety lifecycle represents all safety activities demanded by ISO 26262.

The concept phase is a sequential flow of activities, which allows iterations and exchange of information and results from activities from outside the scope of ISO 26262 (e.g. other technology) and out of the scope of the item.

The ISO 26262, Part 10, should be the basis of the system engineering approach by the SAFE project (according to Figure 6).

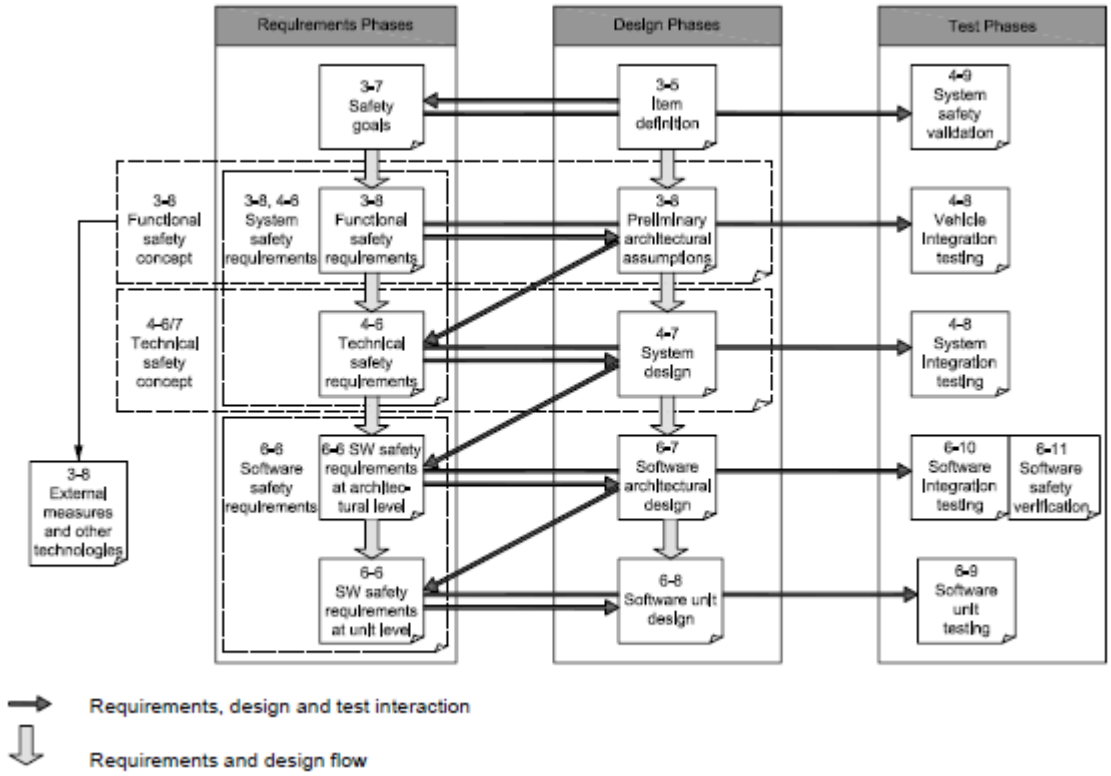

Figure 6: Safety requirements, design and test flow from concept to software (screenshot out of ISO 26262 Part 10, figure 8)

4.2 Concepts

An incomplete list of concepts defined e.g. by founded projects is content of this section.

4.2.1 SPES2020

SPES2020 is a unified framework rather associated with embedded software engineering, as indicated by the name “software platform embedded systems”.

Frankly speaking, the modeling framework consists of viewpoints (requiremental, functional, logical and technical) and layers of abstraction. Two basic engineering approaches for proceeding along viewpoints and abstractions are discussed.

Safety aspects are addressed with a fault-tree-related approach and some WCET-framework in case of federated modular processing

4.3 Methods

An incomplete list of methods defined e.g. by founded projects is content of this section.

4.3.1 AUTOSAR (www.autosar.org)

The method of AUTOSAR is defined as a base for a new method created by the project SAFE.

The following picture is taken from the website of AUTOSAR with detailed information.

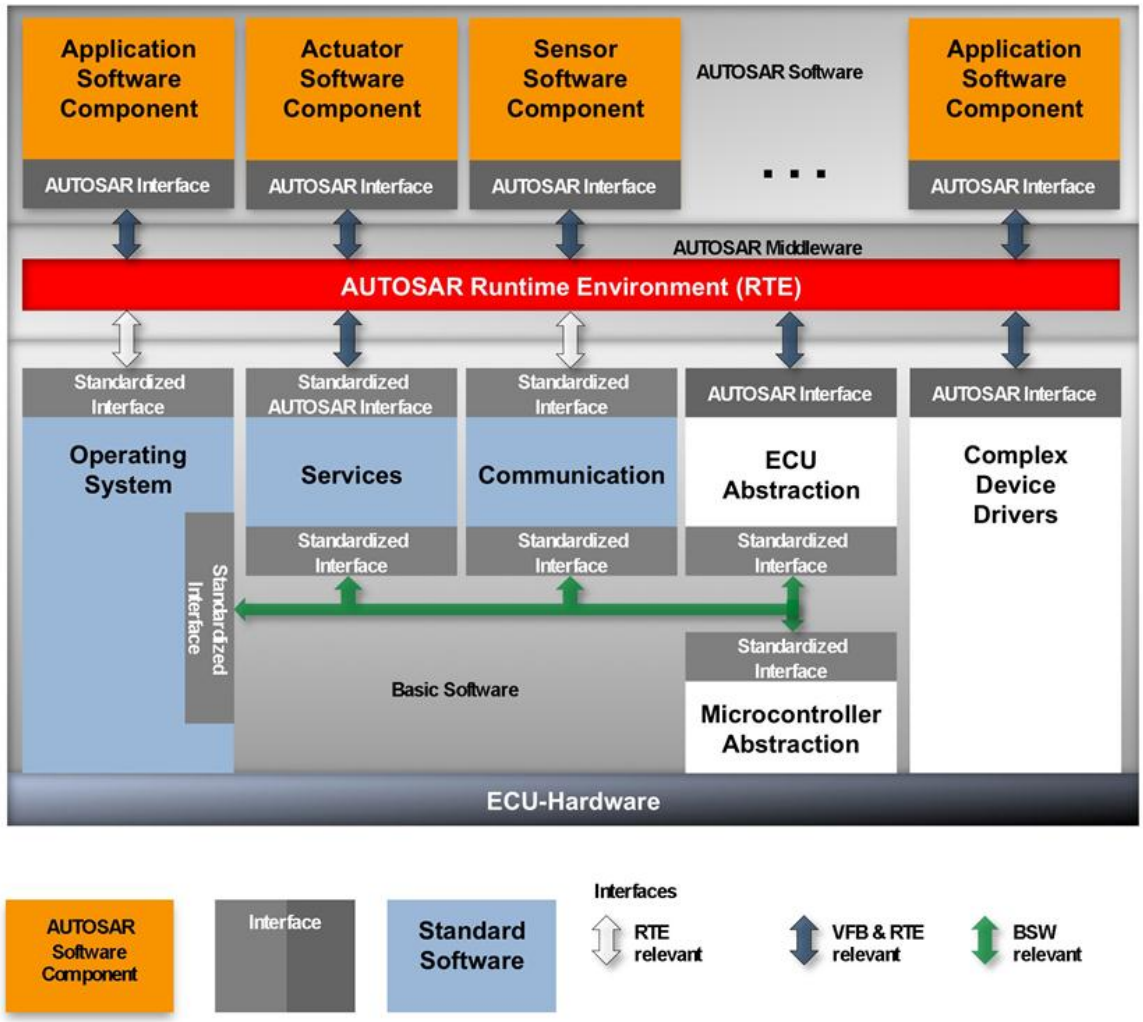

Figure 7: AUTOSAR system architecture

4.3.2 EAST-ADL

This method is defined as a base for a new method created by the project SAFE.

The following information are taken from the web site of the project MAENAD, is part of the project description.

“EAST-ADL is an architecture description language tailored for the automotive industry. The EAST-ADL approach relies on AUTOSAR for representing software architecture but extends to more abstract representations. It includes support for requirements engineering, safety engineering, variability management, and product line architectures.”

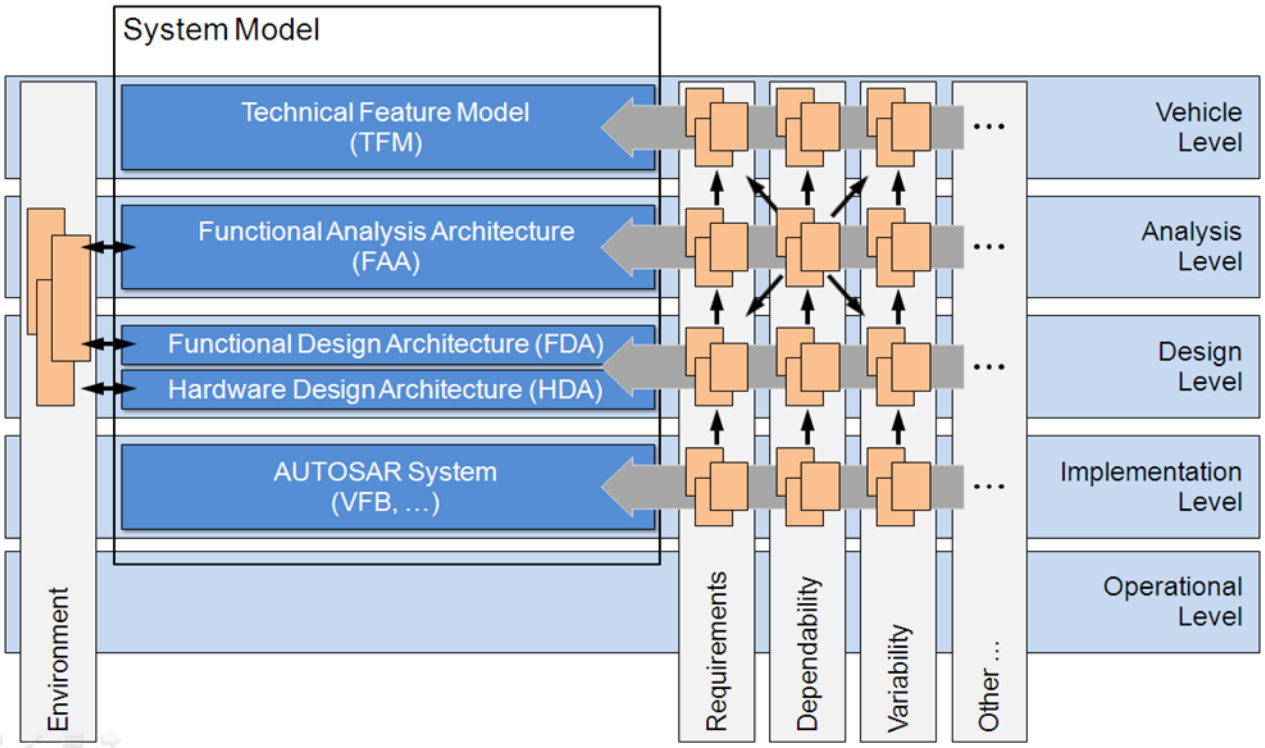

Figure 8: EAST-ADL levels and system model

4.3.3 Model based development methods derived within the SAFE-Project

These methods are derived in the project SAFE taking into account safety requirements. They are derived from the initial gap analysis and the methods for safety analysis, as detailed below, using model based technology as targeted in the project. They are documented within the SAFE meta-model (Enterprise Architect project file).

The following objectives are addressed by these methods with respect to safety process requirement:

- Support for hazard analysis and safety requirement expression and traceability
- Support for Safety case documentation
- Description of automotive architecture with respective system, hardware and software elements necessary to capture the Function Safety Concept, Technical Safety Concept and Hardware and Software safety component
- Description of COTS component
- Model based techniques to support qualitative and quantitative evaluation of safety concept for analysing impact of safety mechanism at different level of abstraction (System, Software and Hardware).
- Model based multi-criteria analysis to benchmark automotive architecture with consideration of safety related element and process
- Capture of formal low level safety requirement to allow automatic code introduction of software safety mechanism in AUTOSAR architecture
- Support of product line and variant selection with safety process in regards to above described objective
- Recommendation to use the AUTOSAR layer and HW resident protection to deploy AUTOSAR architecture for mixed ASIL criticality application (notice that no extra modelling element are defined)

The model based development methods intends to improve existing methods based on existing modelling language such as EAST-ADL and AUTOSAR, and to create safety extension to support and justify safety process related active based on modelling techniques. It do not define the modelling of the process itself in the SAFE meta-model.

4.4 Process descriptions

An incomplete list descriptions of processes defined e.g. by founded projects is content of this section.

4.4.1 EASIS

The project "**E**lectronic **A**rchitecture and **S**ystem Engineering for **I**ntegrated **S**afety Systems" (01.01.2004 – 28.03.2007) was funded by the European Commision.

For the realization of Integrated Safety Systems a powerful and highly dependable in-vehicle electronic architecture and an appropriate development support is mandatory.

The goal of the EASIS project was to define and develop technologies:

A platform for software-based functionality in vehicle electronic systems will be defined providing common services upon which future applications can be built.

A vehicle on-board electronic hardware infrastructure which supports the requirements of integrated safety systems in a cost effective manner will be specified.

Methods and techniques for handling critical dependability-related parts of the development lifecycle will be analyzed, adapted, extended and defined.

An engineering process and a suitable tool chain will be defined, enabling the application of integrated safety systems.

Results of the EASIS project are used for process implementations for the SEP.

4.4.2 MAENAD (http://www.maenad.eu/)

The project "**M**odel-based **A**nalysis & **E**ngineering of **N**ovel **A**rchitectures for **D**ependable Electric Vehicles" is funded by the European Commision.

The following information are taken from the web site of the project MAENAD, is part of the project description.

"The engineering of Fully Electric Vehicles (FEV) introduces new challenges to the automotive industry. Chassis and powertrain systems of FEV will have more authority, be more integrated and rely less on mechanical backup. The complexity and criticality are thus high and rigorous support for complexity management and safety engineering is required.

The MAENAD project continues the refinement of EAST-ADL for meeting these challenges. The title, Model-based Analysis & Engineering of Novel Architectures for Dependable Electric Vehicles gives a hint of the main objectives:

- Provision of support for the automotive safety standard ISO 26262
- Provision of capabilities for prediction of dependability & performance
- Provision of capabilities for design optimization
- Demonstration of project results in a practical electrical vehicle design

in the context of EAST-ADL and Fully Electrical Vehicles."

A used result of MAENAD is the process description with the phases of EAST-ADL and GMP's (generic method patter).

The syntax of the process descriptions of MAENAD and SAFE is BPMN 2.0. SAFE results could be integrated into safety swimlanes e.g. in exported documentations because the “SAFE Engeneering Process” is a detailed description for safety activities also based on MAENAD results.

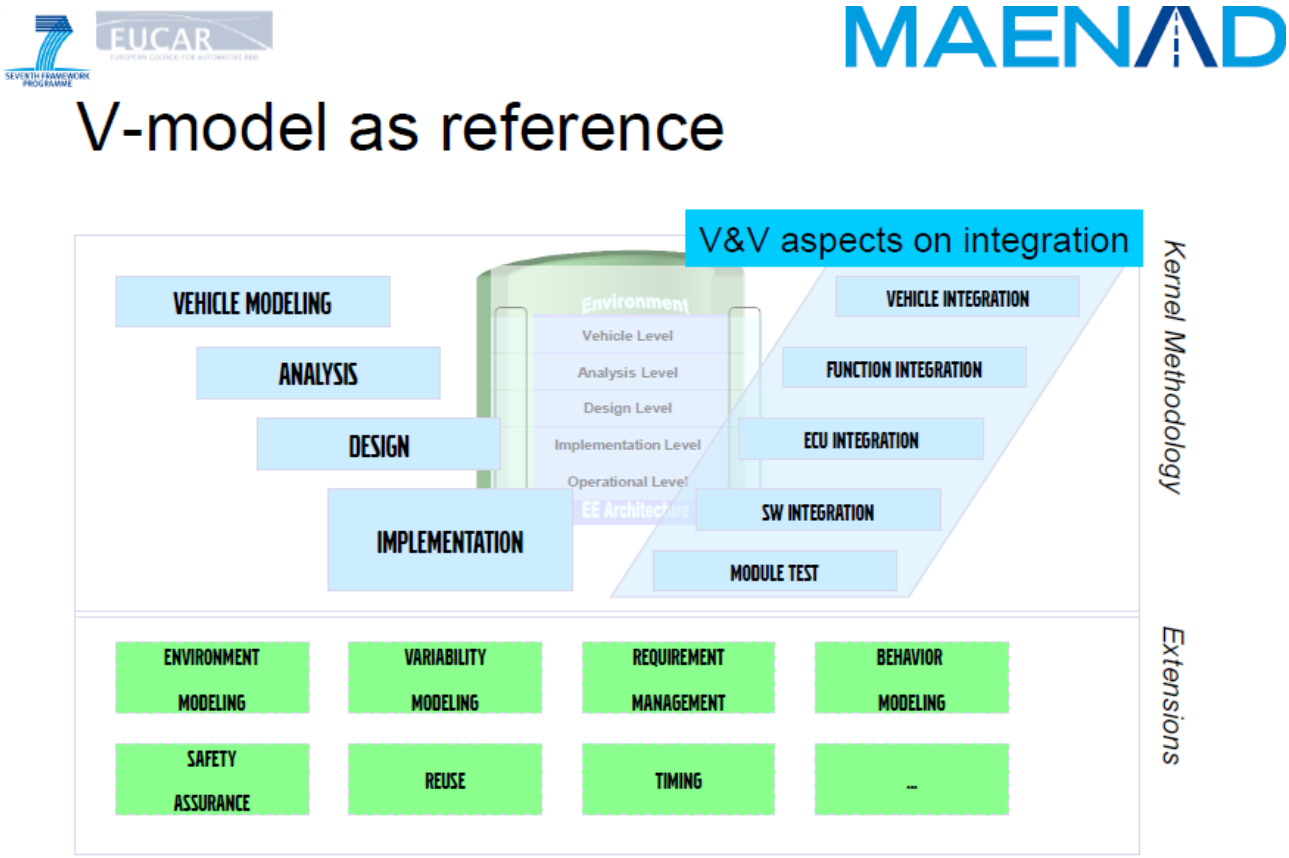

Figure 9: V-model as reference within MAENAD project

5 Safety engineering fundamentals

This chapter gives an overview of a collection of best practices and state-of-the-art in safety engineering of complex systems. This ranges from explanations of development phases according to the ISO26262, the varying architectural abstraction layers required to represent complex systems, and on to the representation of functions, the allocation of requirements, faults and anomalies to them and the analyses of their failures.

5.1 Information as required by ISO 26262

The chapter shows a view on the input / output relation as required by ISO 26262.

ISO 26262 requires the usage of work-products as an input for further activities. In detailing the definition of pictures in Part 10 the following phases of activity had been considered:

- Requirements Phase
- Architecture Phase
- Analyse Phase
- Design Phase
- Verification Phase
- Integration Phase

In all phases of activities information are distributed in horizontal (e.g. from requirement to integration) and vertical (e.g. from vehicle level down to part or unit level) direction.

The requirements and their work-products are from ISO 26262 are referenced in the relevant activity box of the following picture.

Any concept, also software or hardware safety concept (which is not defined in ISO 26262) in the dedicated horizontal level requires activities during the phase:

- Requirements Phase
- Architecture Phase
- Analyse Phase
- Design Phase
- Verification Phase

In the approach of a V-model, it could be considered as a deductive development phase (see also chapter 5.2 in this document) of the descend branch of V-cycle. Integration Activities (ascend branch of V-model) and verifications (e.g. Analysis, Tests, Simulations for requirements or designs etc.) during development (ascend activities in the descend V-branch).

Important is, that the information flow in any horizontal level during development are equal. Any additional system level could be added in between, depending e.g. complexity of the product.

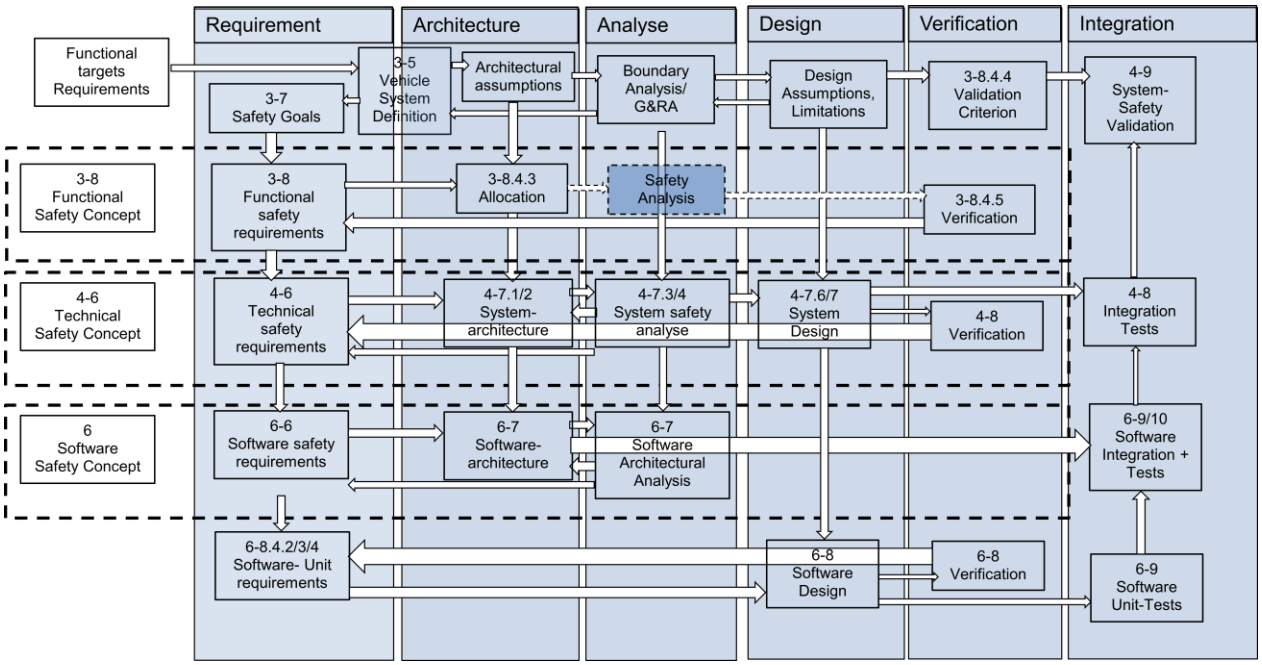

Figure 10: Proposal for Software

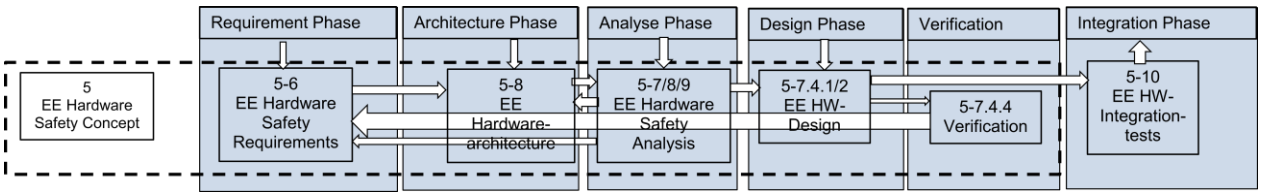

Figure 11: Proposal for EE Hardware enhancement

All such information between the activities shall be made transparent. In order to develop tools to support those activities and their interfaces, a specification of the interfaces is needed.

5.1.1 Architectural views in relation to ISO 26262

The concept required for automotive product decomposition as defined by safety process requirement from ISO26262 leads to introduction two main principles for product representation:

- Abstraction level (refinement): decomposition of the product by refinement according to engineering discipline (e.g. System, software, hardware, mechanics,…). The application of this concept represents a hierarchical design flow.
- perspective: providing a given collection of view point on the product (or its level of abstraction) by considering only a given aspect of the product (control, safety, interface …). The application of this concept represents a horizontal design flow.

The selected reference for product architecture is given from the state of the art SPES architecture (http://spes2020.informatik.tu-muenchen.de/spes_xt-home.html), and from the EAST-ADL (http://www.east-adl.info/).

As depicted in the following figure, the different abstraction defined to represent the automotive product, from item identification to implementation part as a physical element (electronic part, software code, mechanical part, etc..) is organized in relation to ISO2662 definition as :

- A system level decomposed in two abstraction required by the Part4 of the ISO26262. The Functional Abstraction that allow representing the Functional Safety Concept. The Technical Abstraction representing a decomposition of the product for the different physical functional block constituting the product architecture, respectively System block, Hardware block and software block. At this level of design the block represent an abstraction of the

physical component (where further 1:1 or N:1 grouping can be performed thanks to component technology implementation decision). This technical abstraction allows representing the Technical Safety Concept.

- A component level, representing the physical part building the final product as required by Part 5 for hardware and 6 for software in the ISO26262. This component level supports the complete design of the product considering implementation and technology selection, with possible grouping of blocks. This component level is build with software architecture, hardware architecture and system component architecture (mechanics or hydraulics components).

Moreover, the figure also integrates the different perspective required to engineer an automotive product. This list is based on standard design process activity to support Requirement Engineering phase and Architecture Phase, enriched safety perspective as required by the ISO26262 during perform of safety analysis related activities. So finally we have the following detailed perspective:

- Operational Perspective: How e.g. human interact with the product.
- Functional Perspective: How the required function could be experienced or the systems behavior be observed.
- Variability Perspective: How variability is required or could be observed.
- Environment Perspective: Interaction and dependability between product and environment.
- Logical Perspective: Description of product by logical elements.
- Technical Perspective: Description of product by technical elements
- Geometrical Perspective: geometrical positioning of the product.
- ISO 26262 View: Relation of the perspectives to the functional and technical safety concept as required by ISO 26262.

Within perspectives as horizontal design flow and refinement as vertical flow are the support for the standard V cycle largely used in automotive industry. In all horizontal level the different perspectives could be considered. That does not mean that in all defined horizontal level a complete specification of the elements are required.

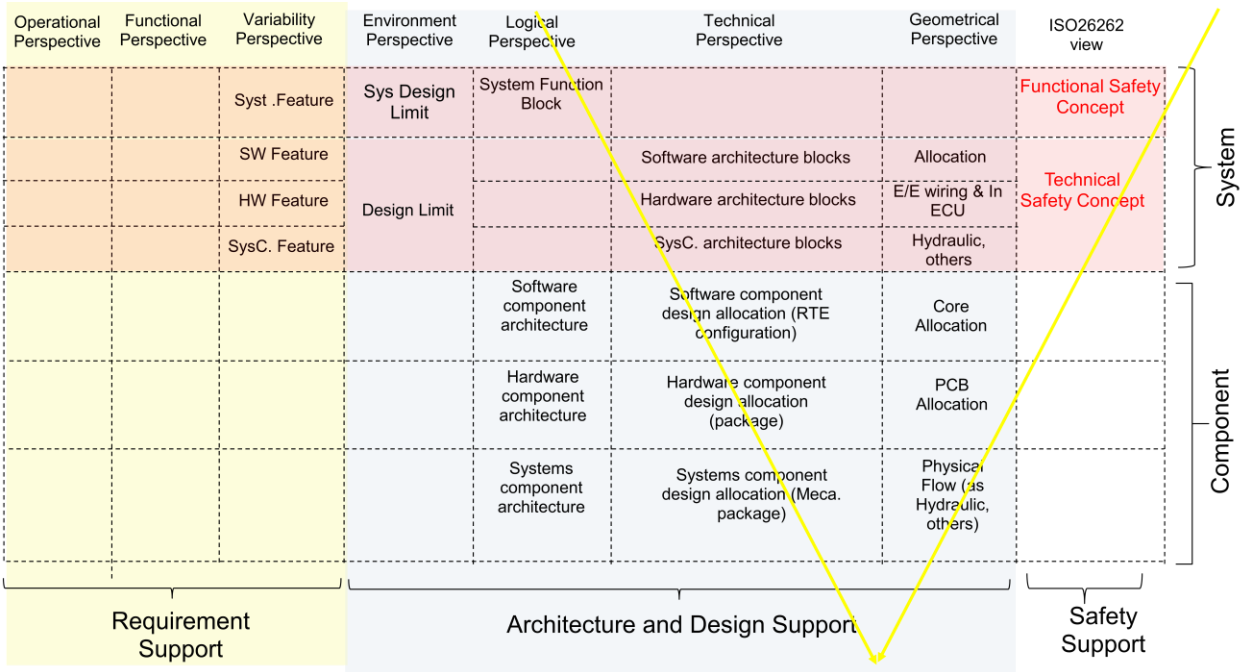

Figure 12: Matrix of perspectives of an item

In order to fulfill the basic requirements from ISO 26262, the Safety perspective shall be engineered with a collection of views (not a perspective like previous picture) required to build the complete picture needed for dependability purpose.

Note that the views can be applied on the different stage of refinement of the product, as defined above. Applying views on abstraction level ensure continuity in the design by decomposition and correlation of analysis results across the abstraction. It permits to validate the assumption or selection defined in the upper abstraction level.

The respective views are (as depicted in Figure below):

- Architecture: represent all the structural element building the architecture as a black box (with respect to abstraction level for element representation)
- Failure: represent the fault or malfunction of the element as properties of the black box. It defines also how fault/failure propagates in the architecture between elements.
- Timing: represent the timing aspect and time propagation of the structural element.
- Functional Behavior: represent the behavior of the element of the architecture as a white box of the element. It is also called positive view, by definition the behavior of the function for what it is intended for.
- Dysfunctional Behavior: represent the failure behavior of the element of the architecture a white box of the element. It is also called negative view, by defining how the fault or malfunction propagates in the element to become a failure.

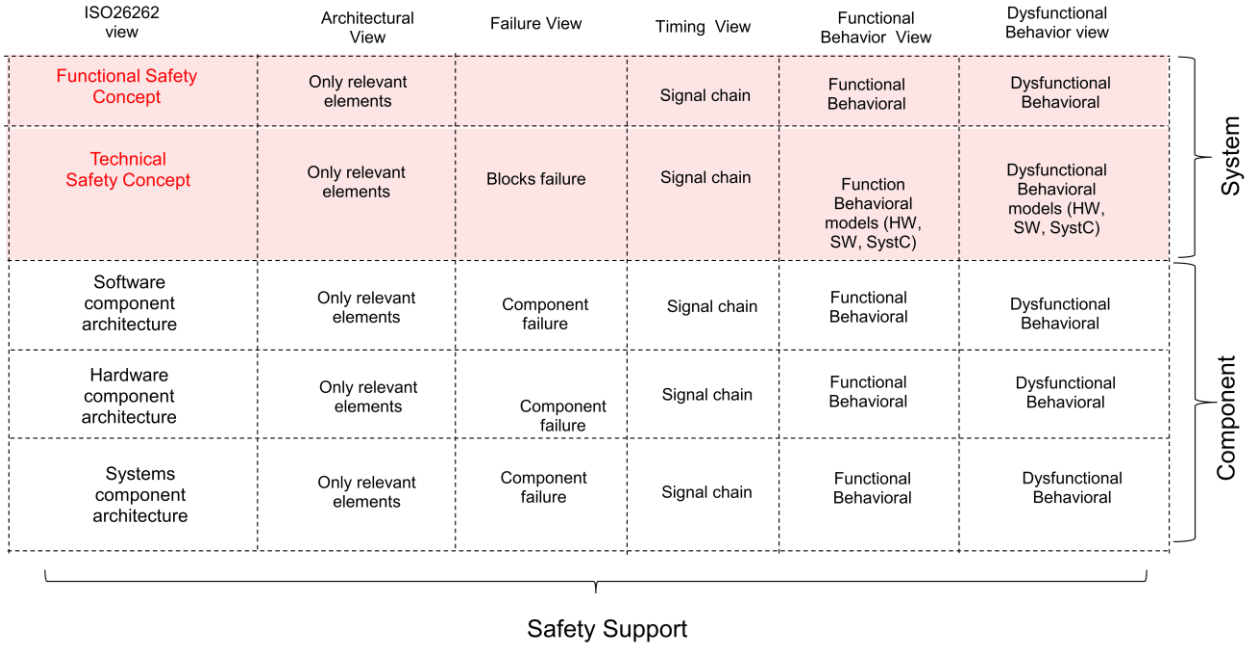

Figure 13: Differentiation between system and component view

Moreover, the system product is strongly influenced by the environment and in particular it is designed to operate safely only in specific range of characteristics of the environment. These Design Limitation Constraints resulting from the environment have significant impact on the correct function of the considered element.

As depicted in the figure below the safety requirement are intended to be defined to express the intended behavior of the system (or its decomposition in the abstraction), but also to tackle the environment limit for the intended design. The resulting properties into the architecture block and environment block has to be captured and traced.

In addition non functional requirement, such as timing, cost, environmental condition, etc may have also impact on the system itself and on the limitation in the environment. These conditions have to be identified and tag as safety related as far as possible, or then refined in safety requirement in order to be able to capture and trace identified properties on the respective architecture block and environment block.

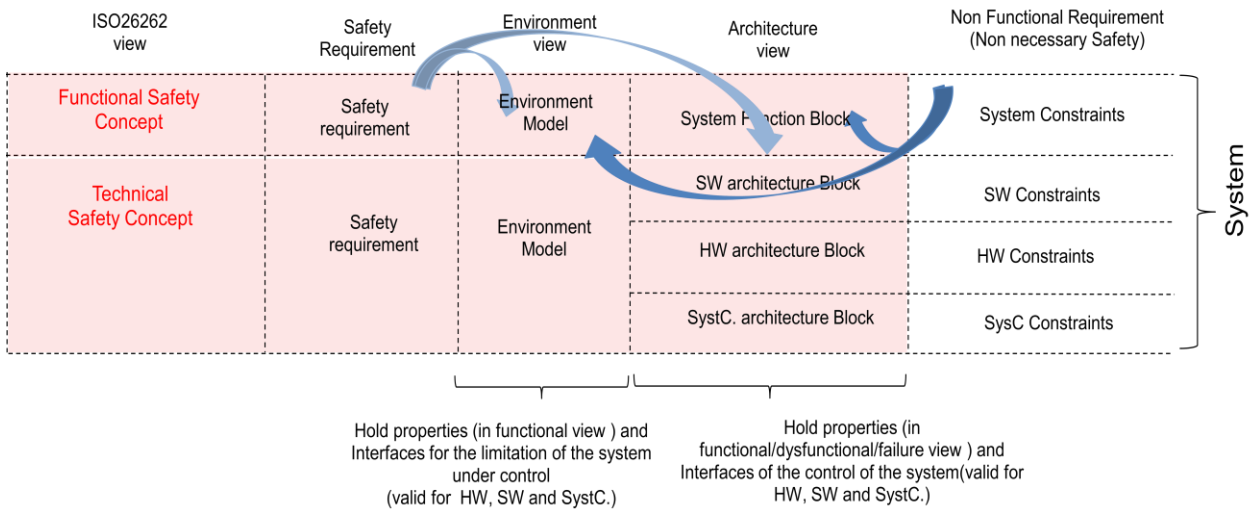

Figure 14: Influence of Design Constraint Limitation

The general concept defined in the figure above with abstraction, perspective and view need to apply with architecture language description selected by the project. While EAST-ADL and AUTOSAR have been selected, the different abstraction and template of these languages need to be applying in particular on Figure 12.

The EAST-ADL modeling representation will be used to represent the System abstraction to represent the Item and theirs different variant from VFM representation (Vehicle Feature Model) and the Functional Safety Concept based on FAA representation (Functional Architecture Analysis). The Technical Safety Concept is representing by the FDA (Functional Design Architecture) and HDA (Hardware Architecture Design) in relation to product variant definition in FM (Feature Model) triggered from VFM variability.

The AUTOSAR template representation will be used for the use of component description in software using Application Software Template and in hardware (and hardware dependent software) using the ECU resource template. The system template is used to defined the integration of the two domain to represent the complete physical implementation and integration of the system.

Note that either in EAST-ADL or in AUTOSAR the System component (Hydraulic or Mechanic component) are representing, as ISO26262 only limit its application to Electronic and Electrical elements.

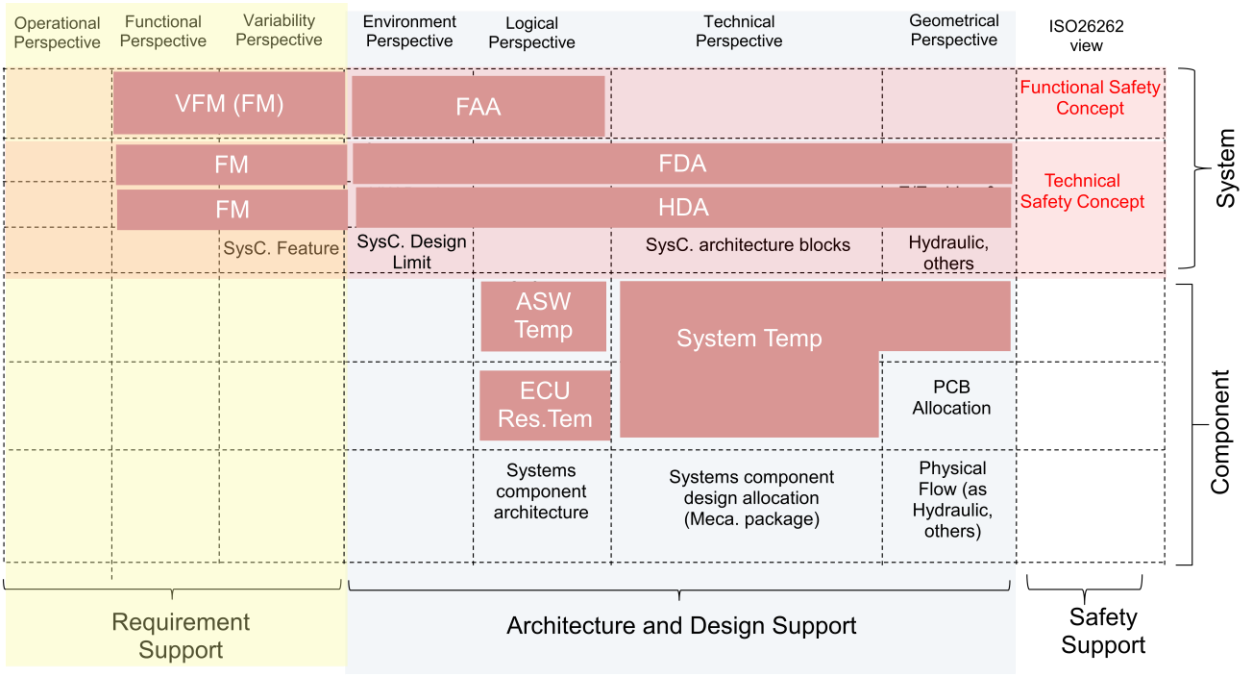

Figure 15: Relation between EAST-ADL / Autosar modeling and architectural views

Finally the system product as represented by EAST-ADL and AUTOSAR language must be mapped to customer supplier value chain in order to support practice commonly used in industry, but also mandatory verification and integration has as defined in the ISO26262 (in Part 4, Chapter 8.1 *"Objective" requires the following integration phases: "The integration and testing phase comprises three phases and two primary goals as described below: the first phase is the integration of the hardware and software of each element that the item comprises. The second phase is the integration of the elements that comprise an item to form a complete system. The third phase is the integration of the item with other systems within a vehicle and with the vehicle itself").*

As a consequence, the systems levels and adequate horizontal levels can be easily mapped to language construct from the above Figure 15.

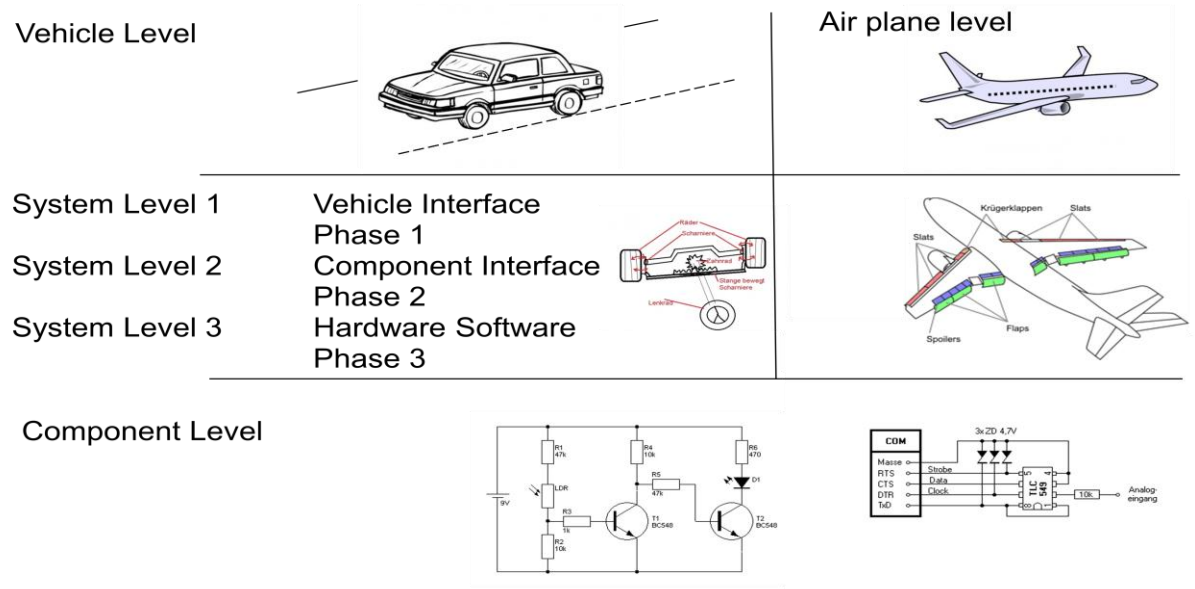

Figure 16: Horizontal system levels

As evidence, the definition of a comparable concept for structuring the item is essential to verify and assess the functional safety of an item.

5.1.2 Representation of functions

When it comes to representing functions and the effect of functional failure, it is highly recommended for the safety designer to clearly differentiate between

1. *Loss of Function* (LF), where the erroneous or advertant (e.g. switch-off) loss of a dedicated function is safety-critical[2] and
2. *Malfunction* (MF), where the erroneous execution of a dedicated function is safety-critical.

The designer should care for LF issues as primary concern owing to their large impact on the physical architecture. All design measures are associated with technical ensurance of the function and should stick to a robust manner, not prone to too many variations during the system development lifecycle. MF issues regarding failsafe design can then follow as second step.

The RAMS (reliability/availability/maintainability/sfety) illustration in Figure 17 below gives the underlying big picture:

- LF (which is associated with the availability part of safety) is at stake for fail-operational missions, where the continued operation has to be ensured owing to Safety reasons. As detection and failsafing is not sufficient here, the safety design[3] has to comprise redundancy mechanisms in order to ensure a default state.
- MF (which is associated with the integrity part of safety) is at stake in stringently fail-safe missions, where cease of operation is the safe state by definition. Here, the classical approach of failure detection and isolation is sufficient[4].

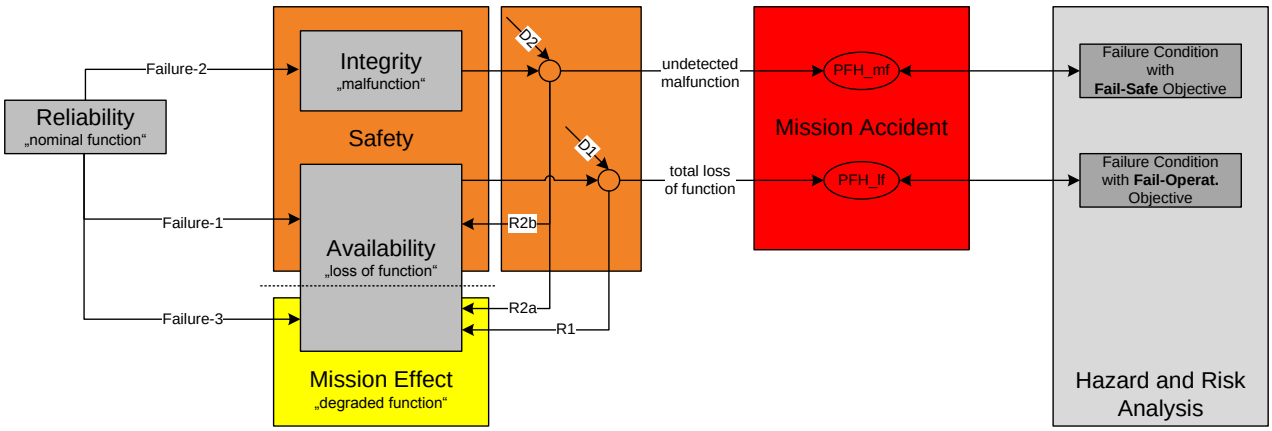

Figure 17: RAMS Considerations

The small description in the figure are as follows:

Priority	Description	Meaning	Design Objective
1	Failure-1	Functional Loss (safety-critical)	• Increase of reliability (e.g. HALT/HASS) • Clear redundancy concept

[2] And hence the mission objective is fail-operational.

[3] RBD is frequently used for LF types.

[4] FTA is frequently used for MF types. Being more general FTA covers LF and MF type analyses.

	R1	Safety Mechanism: Fallback to redundant part	• Redundant design • Ensure safety margin through limitation of degraded mode latency • Ensure control of particular risk
	D1	Detection of functional loss	Ensure safety margin between detection intervals, preferably through passive monitoring
2	Failure-2	Malfunction (safety-critical)	• Increase of reliability (e.g. HALT/HASS) • Establish detection and isolation concept
	R2a (if exist.)	Safety Mechanism: Failsafe Fallback	• Keep track on associated impact (availability up to mission abortion)
	R2b	Safety Mechanism: Fallback to redundant part	• Redundant design • Ensure safety margin through limitation of degraded mode latency • Ensure control of particular risk • Ensure availability of fail-active design
	D2	Detection of malfunction	Ensure diagnostic coverage
3	Failure-3	Functional Loss (non safety-critical)	• Increase of reliability (e.g. HALT/HASS) • Keep track on associated impact (availability up to mission abortion)

A rather practical implication of safety-critical availability (existence of “LF type failure conditions”) is that the design should start with R1 considerations and therefore the technical architecture has to be put in place prior to the logical one.

A functional perspective

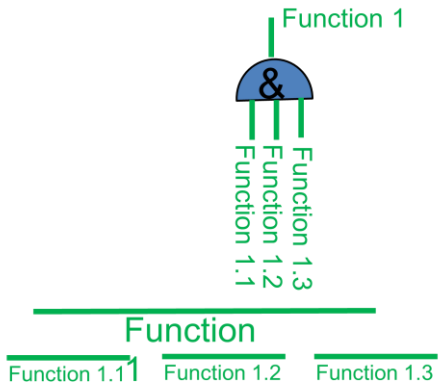

Figure 18: Functional perspective presentation A

Both represent the same result.

Figure 19: Functional perspective presentation B

Function 1 = Function1.1 v Function1.2 v Function1.3

Figure 20: Functional perspective presentation C

Function 1 represents the sequential interaction of the 3 sub-functions.

Those functional perspectives allow no identification of interfaces or boundary. It is limited to describe the behavior.

5.1.3 Identification of malfunctioning behavior using safety analyses

Through the different concept and development phases from the safety lifecycle, ISO26262 recommends or requires, depending on the criticality of the items or elements to be developed, to perform safety analyses.

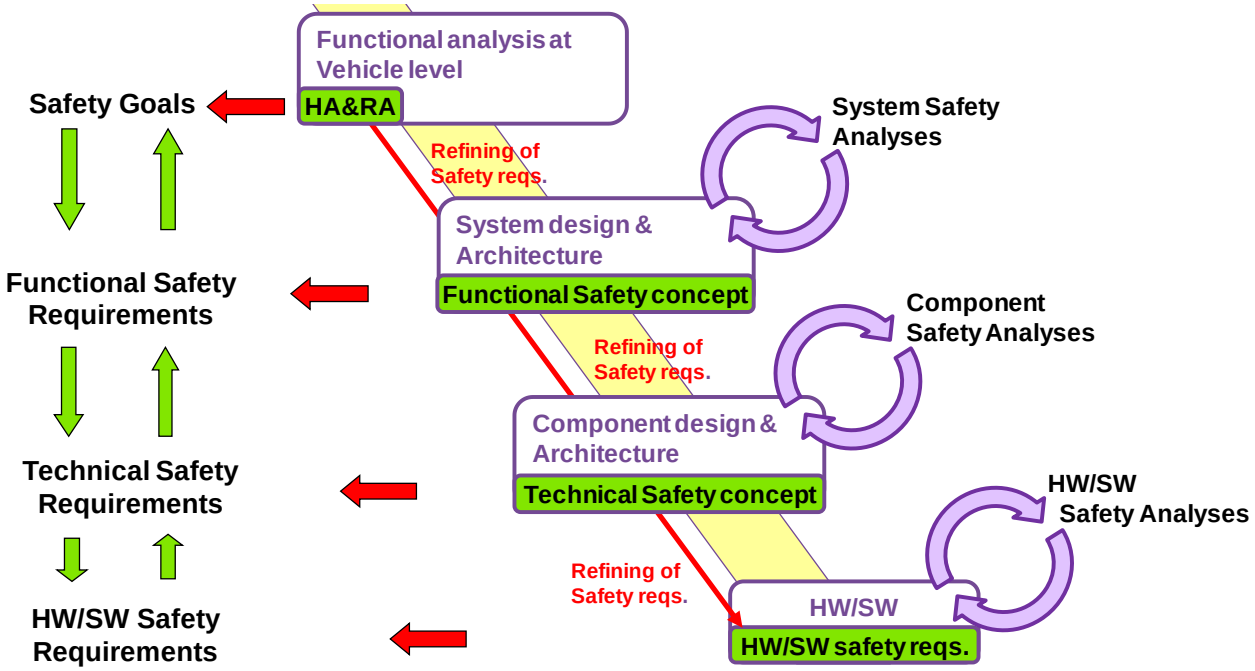

Figure 21: View of safety requirements refinement supported by safety analyses during the concept and development design phases

The objective of safety analyses is to support the derivation of safety requirements from the safety goals, and to validate and verify their effectiveness and completeness.

Safety analyses help to identify the effect of faults and failures on the functions, behavior and design of items or elements. They also provide information on conditions and causes that could

lead to the violation of a safety goal (top-level safety requirement) or a safety requirement. In such a case, additional actions or safety measures shall be determined to eradicate or mitigate the effect of faults and failures.

The fault and failures considered in safety analyses can be either random or systematic, and either internal or external to the items or elements to be developed.

Safety analyses are either inductive or deductive.

- Inductive analysis methods are bottom-up methods that start from known causes and forecast unknown effects. Inductive methods are required by ISO26262 for ASIL A to ASIL D safety goals.
- Deductive analysis methods are top-down methods that start from known effects and seek unknown causes. Deductive methods are required by ISO26262 for ASIL C and ASIL D safety goals and only recommended for ASIL B safety goals.

Safety analyses are qualitative or quantitative:

- Qualitative analyses can be first appropriate and sufficient in most cases to identify failures and when it is not needed to predict the frequency of failure e.g. systematic failures.
- Quantitative analyses extend qualitative safety analyses, in a second step, only when random hardware failures must be predicted as well as the hardware architectural metrics and the evaluation of safety goal violation due to random hardware failures. Quantitative analyses are not required to be applied to systematic failures e.g. software failures.

ISO26262 does not require a specific analysis method but list recognized methods as follows:

Qualitative analysis methods include:	Quantitative analysis methods include:
• Qualitative FMEA[1] (inductive) • Qualitative FTA[2] (deductive) • HAZOP[3](mixed between inductive and deductive) • Qualitative ETA[4] (inductive) • Ishikawa	• Quantitative FMEA[1] (inductive) • Quantitative FTA[2] (deductive) • Quantitative ETA[4] (inductive) • Markov models(inductive) • Reliability Block Diagrams(deductive)
[1]FMEA : **F**ailure **M**ode **E**ffect **A**nalysis [2]FTA : **F**ault **T**ree **A**nalysis [3]HAZOP : **HAZ**ard and **OP**erability analysis [4]ETA : **E**vent **T**ree **A**nalysis	

Table 1 : Example of recognized analyzes methods listed by ISO26262

Additionally, the safety analyses might also contribute to the identification of new functional or non-functional hazards not previously considered during hazard analysis and risk assessment.

For more informations on this topic please read carefully the deliverable D.3.3.1.b under https://itea3.org/project/workpackage/document/download/1563/10039-SAFE-WP-3-SAFED331b.pdf

5.1.4 Error propagation

As largely described in D331a deliverable, the concept for error modeling and error propagation is to extend the architecture structural elements of automotive on each abstraction level, by a relation for component error annotation organized in an “error model” and in a description of its malfunctions. The “error model” describes the black-box view in terms of error propagation for the referenced structural element. Thus, “error model” visible interface as “external faults” for incoming faults and “external failures” for output failure can be described.

Error model is also associated with white-box information as the error behavior of an element. In this case, the internal details of the structural element are known and respective “internal faults” as well as process faults” can be described. In addition, it is possible to describe how “malfunctions” as “external faults”, “internal faults” and “process faults” are related with “external failures”, or with other words: how do those faults contribute to the unintended behavior of the architectural element associated via the error model. This behavior is defined as “error behavior”.

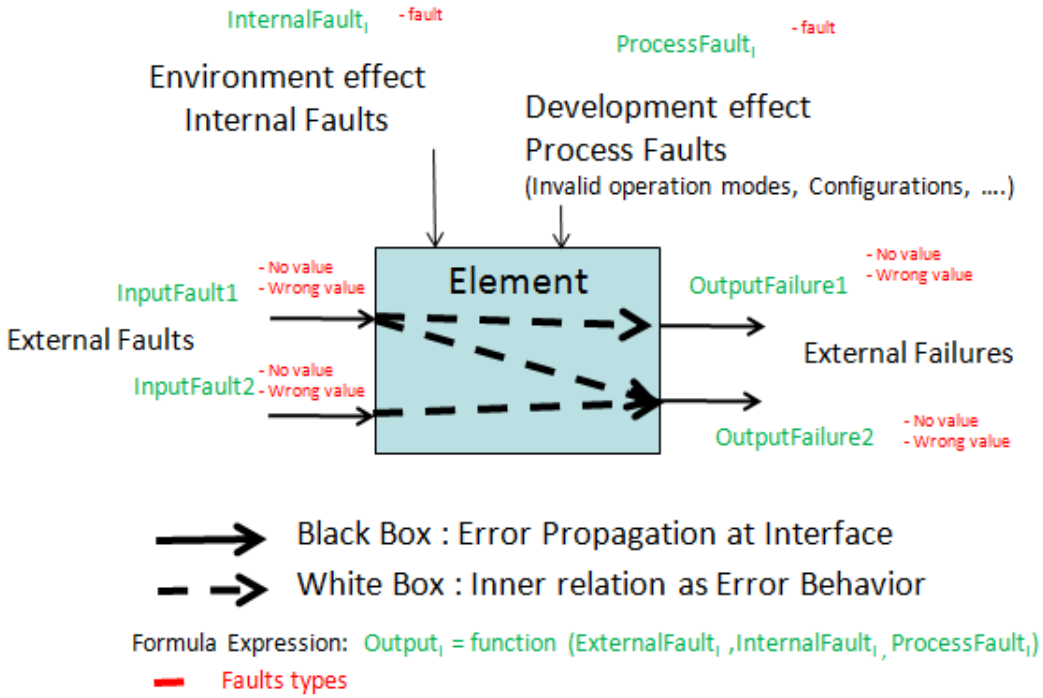

Figure 22: Error model concept

Moreover, the integrated view of the error models allows defining the overall concept for error propagation. Here we may distinguish two layers for error propagation: horizontal and vertical.

The horizontal error propagation does not cross layers of abstraction of the system architecture, but it follows any potential dependability in the same abstraction. Such as errors flows through the functions and energy nets as well as any dependability related to common use of resources. This propagation allows dependability analysis by connecting error model of the same abstraction layer, typically function to function, software to software or hardware to hardware. Moreover the hierarchical modeling approach, typically software component and execution infrastructure or software infrastructure and hardware execution unit, it is then possible to describe, how external faults can be caused from preceding architectural elements. Such approach allows introducing built-in barriers (safety mechanism) that could hide the potential error propagation.

From the above error model description, “external failures” and “external faults” are connected via the support concept the cause-effect relation. Malfunction from functional elements, or from components either from hardware or software domain, propagates via a “fault failure propagation link”. In the hierarchical approach, external faults can be caused from preceding architectural elements (e.g. communication partner, execution environment, hardware software interface) the same constructs are used. This way it is possible to describe a complete error propagation chain from the root fault(s) towards the failure of interest.

The vertical error propagation allows handling relation between abstraction level for malfunction, e.g. “internal faults” or “process faults” decomposition in the abstraction and relationship. This vertical propagation, respectively constructed by an “error model mapping” modeling element, permits to correlate the effect of a dedicated internal fault at a low level of implementation compare

to “malfunction” assumption made during safety analysis of the upper abstraction level. The flow of the malfunction path as a vertical net allows tracking the malfunction across the refinement of the model, and to guarantee and justify the control of theirs effects.

These above enounced concepts are integrated in the SAFE Meta model for error model package.

5.1.5 Analysis of dependent failures between Functions and their Realization

Basis for the Analysis of Dependant Failures in ISO 26262, Part 9, chapter 7 is the identification of common error causes and enablers leading to

1. Corruption the fault isolation concept by inadvertant failure propagation (cascade)
2. Corruption of the functional or architectural independence concept through common modes of redundantly used parts
3. Corruption of the functional or architectural independence concept through common causes external to the system

Cascades may e.g. arise from missing series elements. If they exist, they should be accepted. Common modes and causes are redundancy breakers. Common modes may arise if the same element is redundantly used and one of its failure modes may thereby lead to multiple effects. This might be acceptable but should then be appropriately stated. Common causes arise from abnormal external exposures as HIRF, high energy, environmentals etc. If a vehicle crash is e.g. likely to cause damage in functions for which independence is claimed, this should be accepted.

Target of the analysis of dependent failure based on the architecture is to identify the relevant error enabler.

Any dependability related to robust design could not be evaluated by analysis of the architecture. The realized product, a simulation of the realized design or a model which is completely validated versus the design is compulsory.

The figure below illustrates possible examples:

1. A missing isolation device might result in a short-to-ground cascade from A1 to B (and in this dedicated cased event reverse back to A2).
2. A common mode of the parts redundantly used in A1/A2 might lead to entire loss of sensor acquisition
3. Abnormal conditions as fluid intrusion or high EMC exposure might corrupt both S1 and S2 values and thereby render voting approaches ineffective.

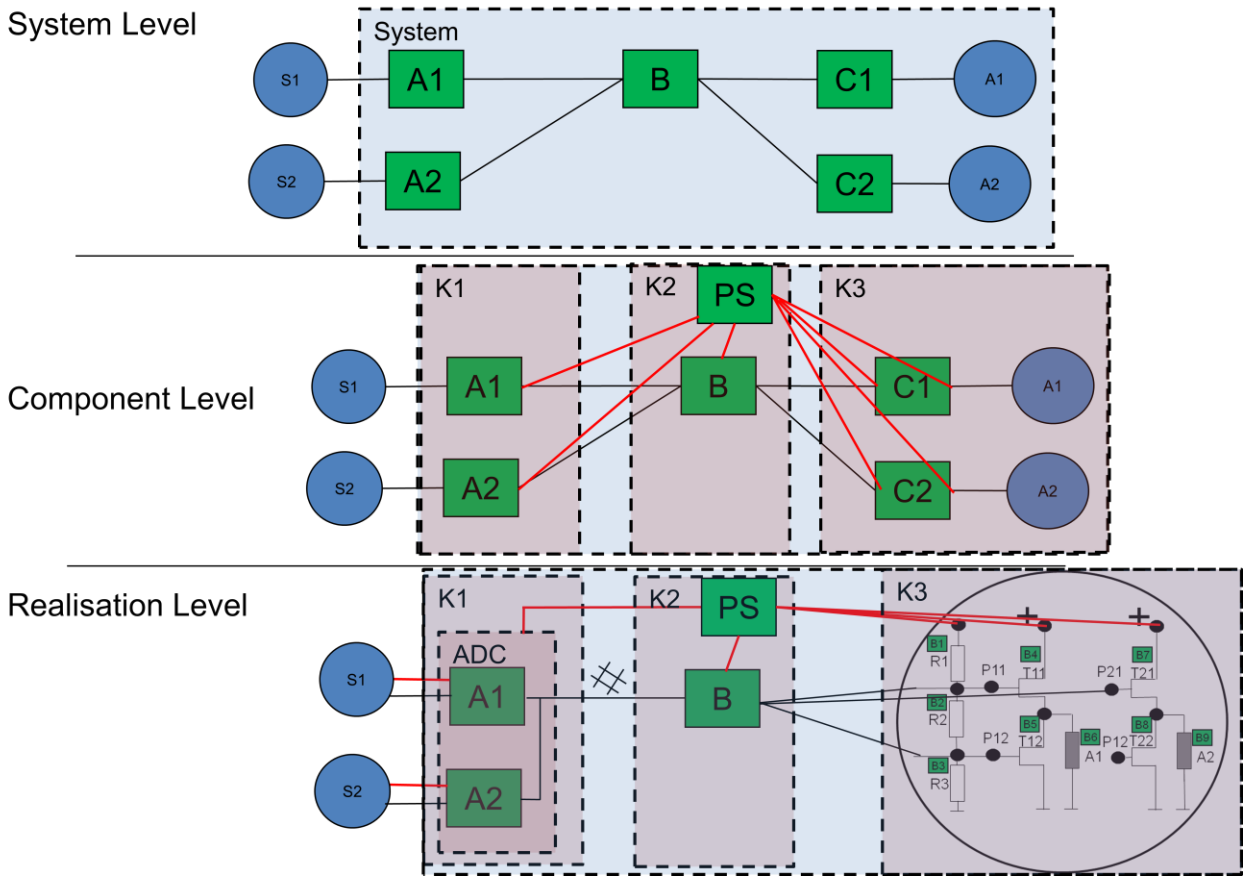

Figure 23: Same function represented on different horizontal level of abstraction

5.2 Deductive Analyses for Proof of Safety Architecture

5.2.1 Background

ISO26262 requires deductive analyses for aligning the overall system architecture with the safety goals derived from initial hazard assessment. This task is crucial for the entire system design, as conceptual proof of the safety (and also reliability, availability, maintainability and serviceability – see RAMS) philosophy.

In early development stages, when no detailed design is known, it is – besides clearly stating the external boundaries and the internal decomposition of the item - important to specify

1. The ASILs of the subsystems (within the given architecture)
2. The critical failure conditions of the subsystems and their average probability of occurrence and/or existence

The following guidance shall give some hints on how to apply deduction in order to specify these two mandatory.

It is important to emphasize that detailed subsystem design data is neither available nor needed for deductive analysis. The figure below summarizes the Safety activities mandated by ISO26262 along the well-known V-development Model.

With available Hazard potential and rating, the synthetically, “deductive” left part (prior to product development) sets the framework for the entire safety case, as the generated and validated requirements are finally counterchecked by the product-oriented verification tasks (with inductive analysis being one of several means of compliance).

As indicated in the next figure below the deductive analyses are part of “proof of concept”, during which it is to demonstrate HOW System Safety is ensured with the intended systems. The proof should be given prior to critical design freeze. After system implementation it is then part of the verification to prove that System Safety is ensured.

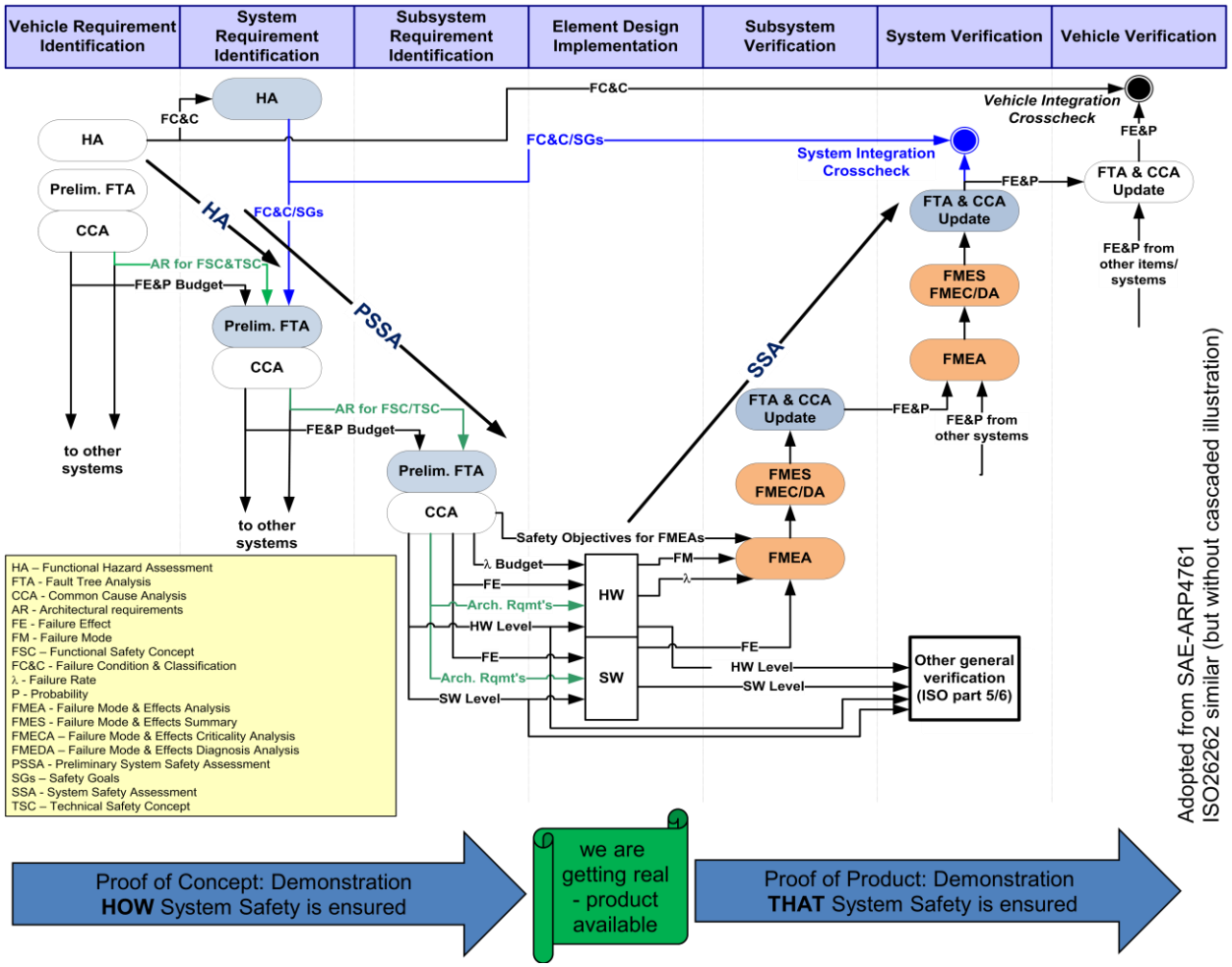

Figure 24: Safety Activites along V-Model

The terminology “Preliminary System Safety Analysis” (adopted from the airborne SAE-ARP4761 standard [15]) comprises the two main activities in the deduction domain:

1. Preliminary Fault Tree Analysis (FTA), in place for deducing the failure conditions leading the “feared events” (which are nothing more than the critical failure conditions mandated by the hazard assessment).

2. Common Cause Analysis (CCA), in place for establishing awareness for the common causes inside and outside the system and deriving appropriate segregation and separation requirements.

Although FTA is named above, ISO26262 does not require to specifically perform a FTA. However any kind of deductive analysis is requested.

In practice, it has turned out to be beneficial to establish

- A Functional Failure Set (FFS) in order to clearly attribute ASILs to the involved subsystems
- A Fault Tree Analysis (FTA) to identify and quantify the failure conditions of the involved subsystems.

It is possible to combine both methods in one representation. However, one objective of this induction stage must be very clear to the involved system designers and analysts:

1. The ASIL allocation addresses design errors (e.g. systematic procedural flaws) and determines the level of rigor to be sustained throughout design, development and verification.

2. The failure condition identification addresses technical failure (e.g. random hardware failure, operator misuse, etc.) and details the conditions for which residual risk of occurrence has to be controlled.

5.2.2 Induction vs. Deduction

The most general distinction between induction and deduction is given by NUREG-0492:

> Induction constitutes reasoning from individual cases to a general conclusion. If, in the consideration of a certain system, we postulate a particular fault or initiating condition and attempt to ascertain the effect of that fault or condition on system operation, we are constructing an inductive system analysis. Thus, we might inquire into how the loss of some specified control surface affects the flight of an airplane or into how the elimination of some item in the budget affects the overall operation of a school district. We might also inquire how the non-insertion of given control rods affects a scram system's performance or how a given initiating event, such as a pipe rupture, affects plant safety.
>
> Many approaches to inductive system analysis have been developed and we shall devote Chapter II to a discussion of the most important among them. Examples of this method are: Preliminary Hazards Analysis (PHA), Failure Mode and Effect Analysis (FMEA), Failure Mode Effect and Criticality Analysis (FMECA), Fault Hazard Analysis (FHA), and Event Tree Analysis.
>
> To repeat–in an inductive approach, we assume some possible component condition or initiating event and try to determine the corresponding effect on the overall system.

Figure 25: Deduction according to NUREG-0492

As a matter of fact, the base for induction is rather small during early stages of system definition, as no detailed design is so far available.

Despite of this, it is crucial to "frontload" the right and appropriate set of requirements, based upon the preliminary system architecture. This is the vital contribution of deductive means. It is important to figuratively keep in mind that

1. Deduction raises the questions to be answered by induction
2. Deduction alone cannot prove Safety – it is the first half of the game only

For final accomplishment of the safety case, both kinds of analyses are combined in a reasonable manner.

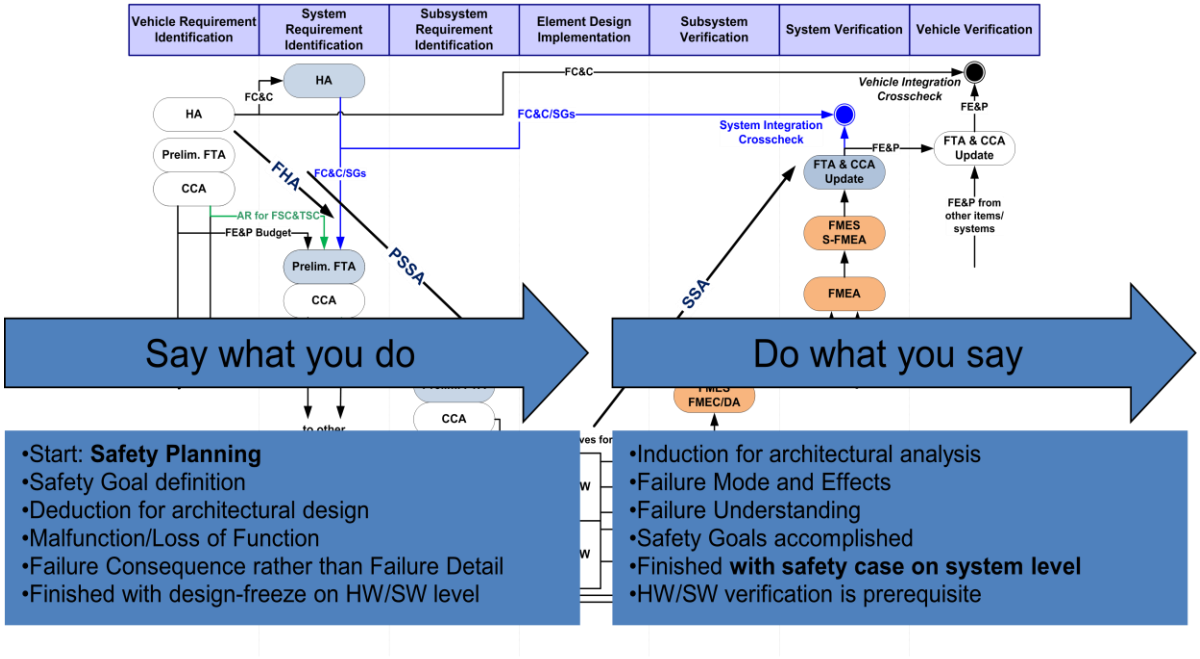

Figure 26: Deduction and Induction Interaction

5.2.3 ASIL via Functional Failure Set (adopted from SAE-ARP4754)

A systematic approach to assigning ASILs, when considering system architectures, is the concept of Functional Failure Sets (FFS).

A FFS is a single element or a specific group of elements that are considered to be independent from one another (not necessarily limited to one system) that lead(s) to a top level failure condition. Conceptually, for ASIL assignment purposes, a FFS is equivalent to a fault tree minimal cut set, whose members represent the result of potential development errors rather than failures. A failure condition may have a single or multiple FFSs. Each failure condition has its own set of FFSs.

If a FFS has one member, the ASIL of this member is the highest of the failure conditions the FFS belongs to. If a FFS has multiple members, ASIL decomposition techniques may be applied.

The FFSs for a given failure condition may be identified by using qualitative safety assessment techniques, such as Fault Tree Analysis or Dependence Diagrams.

The most significant issue is the independence between the elements of one FFS, which must unambiguously be demonstrated. In practice, a simple but well-structured table-based approach is by far more promising than an out blown minimum cut set extracted from an overloaded fault tree.

The FFS analysis outcome has to be specified in the subsystem requirement specification in order to enforce the appropriate level of rigor on subsystem level. The picture below shows an example using MS Excel – the system input and the subsystem requirements are maintained in the MKS RM-based system specification.

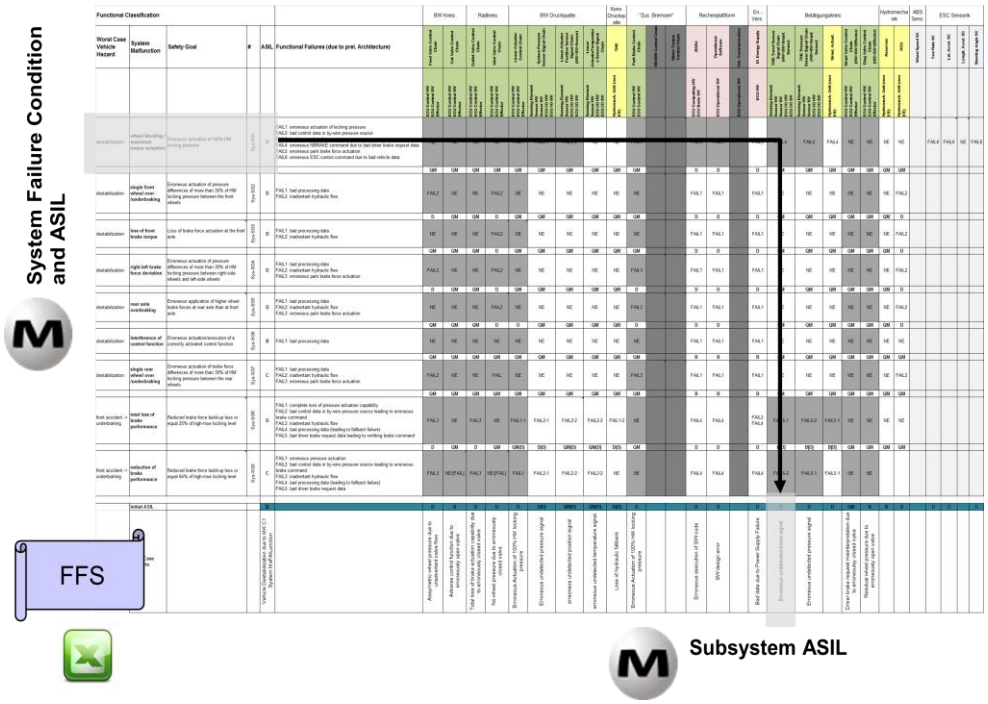

Figure 27: ASIL Derivation

The left side represents the requirement input – the failure conditions from hazard assessment and the ASIL assigned to them.

The bottom line is a part of the FFS outcome – the ASILs of a dedicated subsystem as part of the intended system architecture.

As direct implication of the allocated, the metrical values SPFM (single point fault metrics) and LFM (latent fault metrics) indicate the isolation capability of the subsystem regarding single faults and latent faults.

Note that the subsystem itself is evaluated only on black-box level, with not detailed design data available.

5.2.4 Critical Failure Condition via Fault Tree Analysis

A systematic approach for identifying and quantifying critical failure conditions is the Fault Tree Analysis, which is well-established in several industries with safety-critical background. The eldest and maybe most comprehensive guideline is NUREG-0492 [17].

In a nutshell, the FTA focuses on one (and only one) failure condition (“undesired event”) and provides means to determine its causes. The fault tree itself is a graphical model of various parallel and sequential fault combinations that will result in the occurrence of the undesired event. The fault tree is not a model of all possible system failures or all possible causes leading to system failure – the faults are e.g. not exhaustive, they only cover the most credible ones as assessed by the analyst.

The most powerful mean is the minimum cut set representation, from which all possible causes of the undesired event (as derived from the fault tree) are summarized.

In a top-down fashion, the derived causes represent the failure conditions on subsystem level and thereby constitute qualitative and budgeted (quantitative) safety requirements.

Again it is crucial to fix the FTA outcome in the subsystem requirement specification in order to enforce the appropriate level of rigor on subsystem level. The picture below (adopted from the WP6 Use Case of WT522) shows an example using Isograph Reliability Workbench – the system input and the subsystem requirements are maintained in the MKS RM-based system specification.

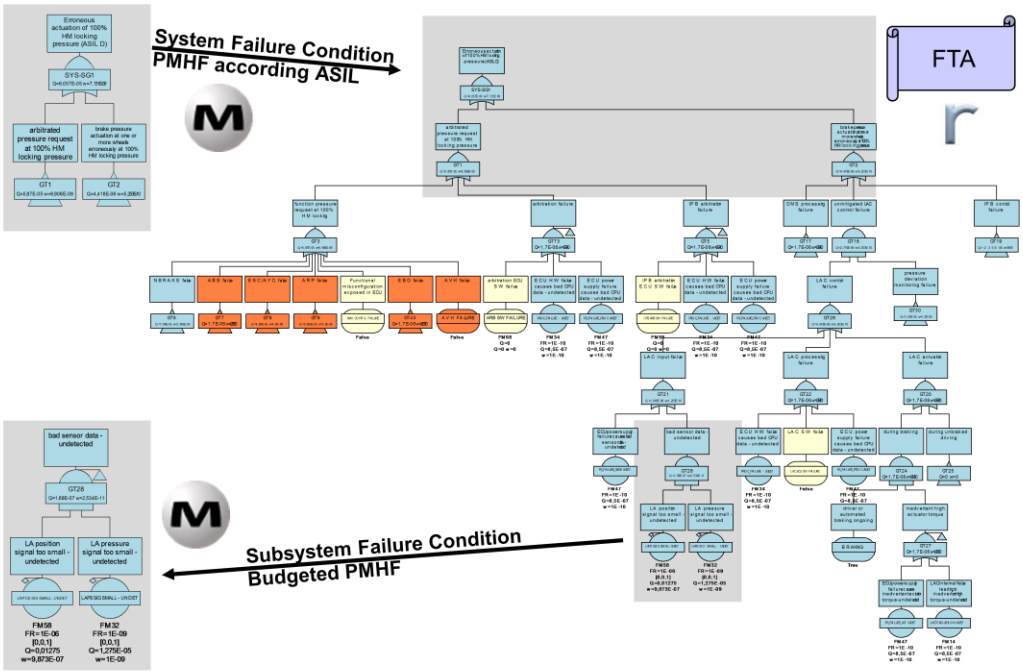

Figure 28: Failure Condition Derivation (adopted from the WP6 Use Case of WT522)

The top left corner represents the requirement input – the failure conditions from hazard assessment and their PMHF as indicated by the ASIL assigned to them.

The bottom left corner is a part of the FTA outcome – the failure conditions of a dedicated subsystem and the budgeted PMHF.

Note again that the subsystem itself is evaluated only on black-box level, with no detailed design data available.

5.2.5 Safety Case Contribution

The difference between induction and deduction finally becomes visible prior to Safety Case closure, as summarized in Figure 29.

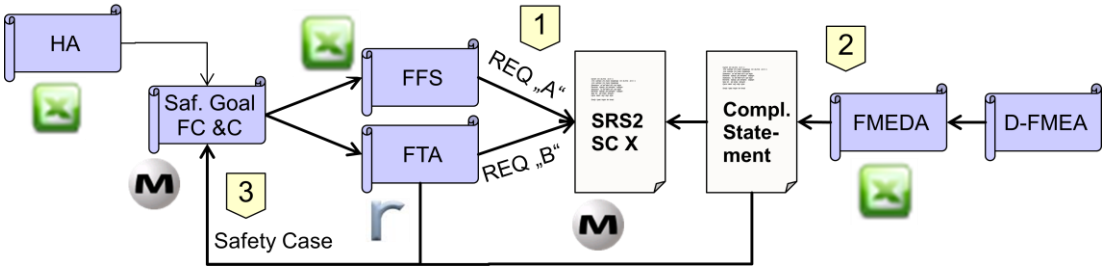

Figure 29: Safety Demonstration

Simply speaking, the questions asked by deduction have, let alone being correct and valid themselves, to be answered by induction, as summarized by the following table:

1	Architectural Requirement Derivation • „A“: Subsystem ASIL, SPFM/LFM Targets, Segregation Constraints (via FFS) • „B“: Critical Subsystem Failure Conditions and PMHF Targets (via FTA)
2	Compliance Demonstration • Design Review • Analyses
3	Safety Demonstration 1. Requirements are valid (e.g. appropriate in terms of hazard mitigation) 2. Design solution is compliant with requirements

Table 1: Three analytical steps

Note that a FMEA itself is not sufficient as it does not provide any statements on criticality and isolation capability of the identified failure effects. Therefore a respective expansion must be provided, which is referred to as FMEDA, associated dedicated FMEA data to the FTA based events. As a seamless approach is not feasible, this process has to be governed by requirements.

The final third step finally yields the desired contribution to the Safety Case.

6 Guidelines and activities in the ISO 26262 Concept Phase

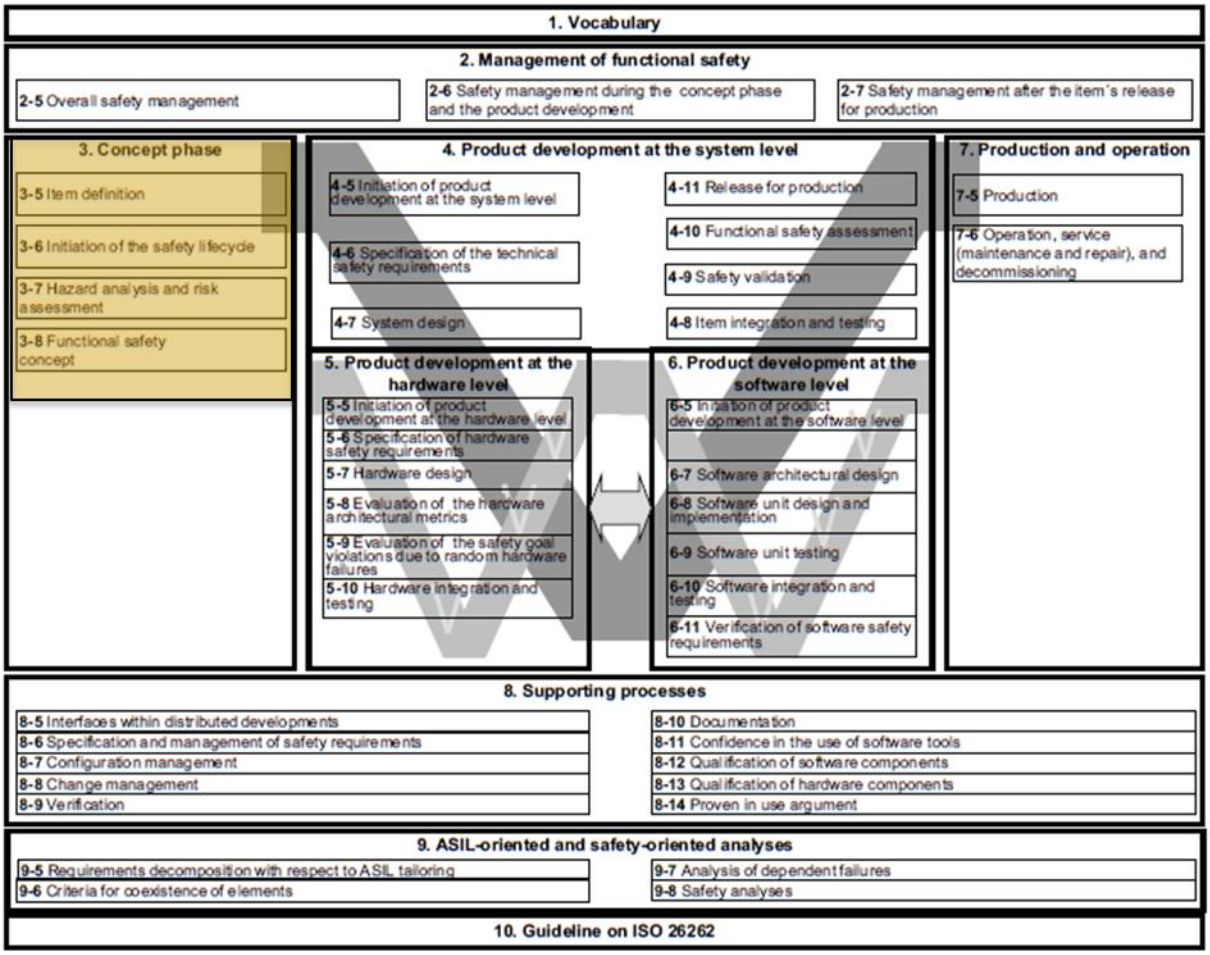

Figure 30: ISO 26262 Concept Phase

6.1 General description of the phase

The concept phase is covered by ISO 26262 Part3.

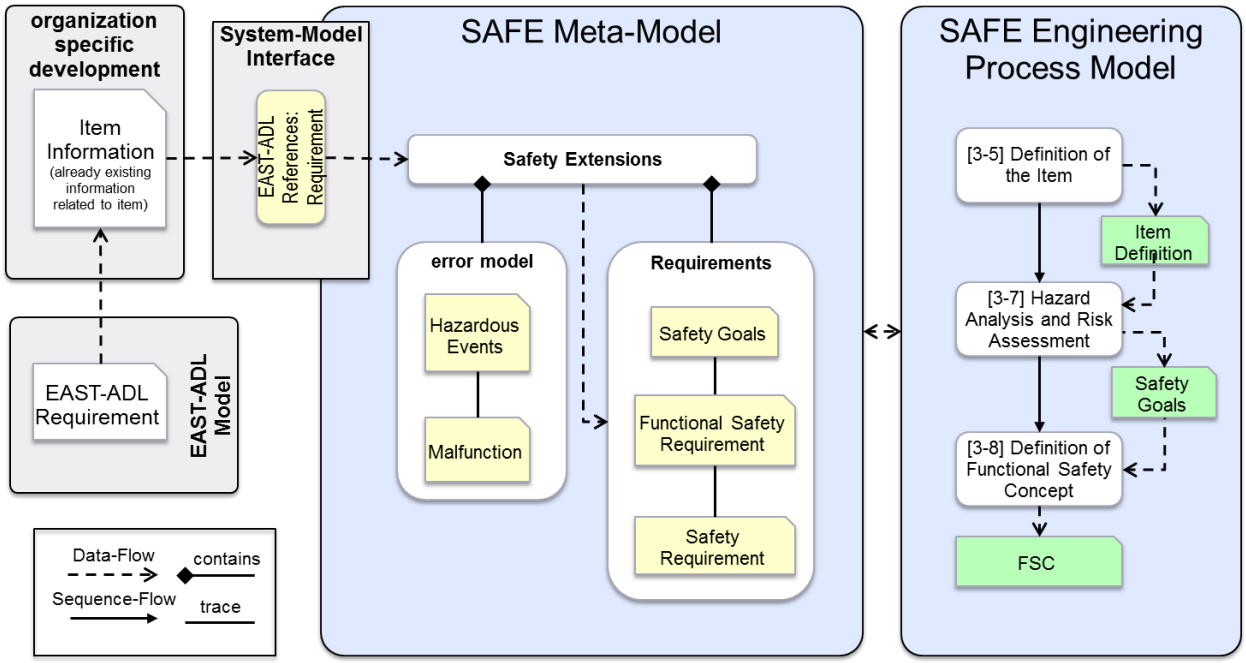

Figure 31: Concept Phase Overview

6.2 Item Definition

6.2.1 Activities

This activity support an adequate understanding of the item so that the activities in subsequent phases can be performed.

This activity is covered in the SAFE Engineering Process (SEP) Model. Further details see 6.2.3.

6.2.2 Formalism in Meta-model

The Item Definition shall describe the item, its dependencies on, and interaction with, the environment and other items. Furthermore it shall support an adequate understanding of the item so that the development oft he item can be performed effectively.

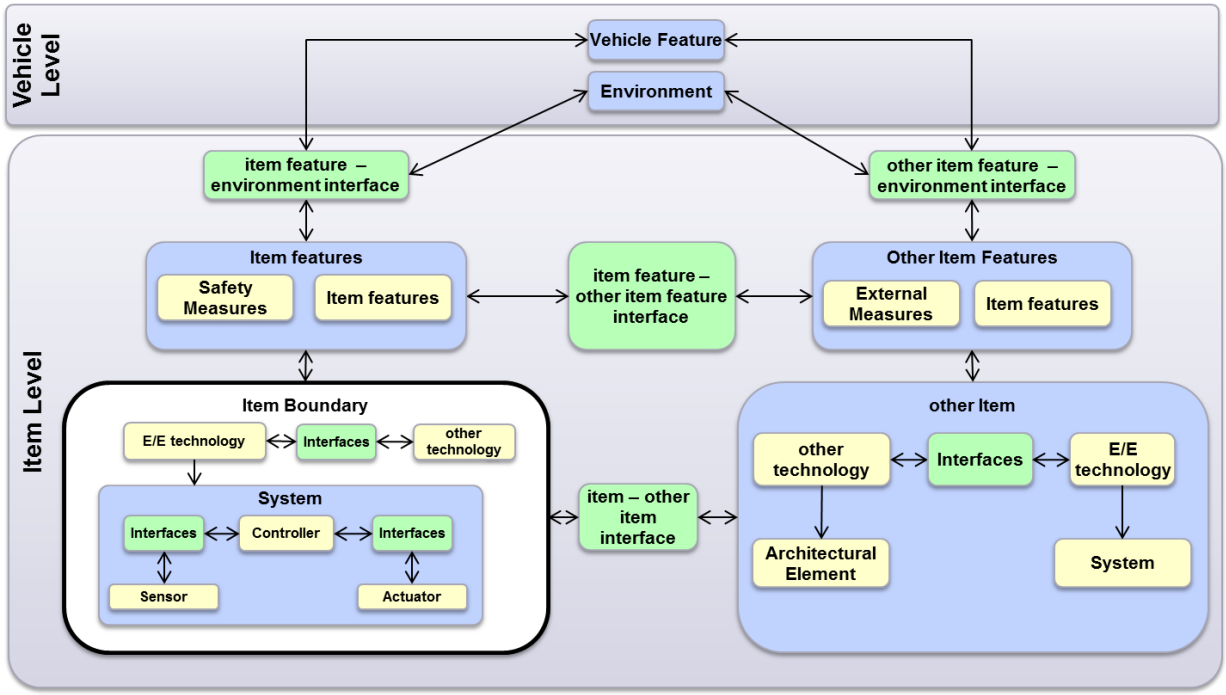

Figure 32: Item Interfaces

6.2.3 Formalism in SEP

The following activities are modelled in a EnterpriseArchitekt model to represent the “Item Definition” topics.

“[3-5] Definition of the Item” (see chapter 13) in the swimlane “requirements engineering”

“[3-5a] define preliminary architectural assumptions” (see chapter 13) in the swimlane “architecture engineering”

6.2.4 Exemplary realizations in tools

Model-based item definition tools are not currently in wide usage in the automotive domain.

Based on common practices in the automotive domain, item definitions are usually generated by the usage of rich-text capable document processing tools.

6.3 Initiation of Safety Lifecycle

The ISO26262 specifies the starting of „Initiation of Safety Life Cycle“as explained in Part 3, Chapter 6. The ISO 26262 does not prescribe any certain process model, but requires the existence of rules and processes, complying with the standard’s requirements, as stated in Part 2.

As per the scoping definition of the SAFE project, process related tasks and activities are not within its scope and focus. Nevertheless, some documentation is supported in the SAFE Meta-model using the safety case capability. Moreover, the SAFE Engineering Process (SEP), which maps the activities and artifacts supported by the SAFE Meta-model onto a generic process model, can be used to support the Safety Life Cycle, as defined by the ISO26262. There is however no explicit support for the Initiation of Safety Life Cycle, as defined by ISO26262.

6.3.1 Activities

Initiation of safety lifecycle shall be used to make the distinction between a new item development and a modification to an existing item.

Furthermore the safety lifecycle activities that will be carried out in the case of a modification shall be defined during this activity.

6.3.2 Formalism in Meta-model

Process activities are not part of the SAFE meta model.

6.3.3 Formalism in SEP

This chapter will be filled in D6.b.

6.3.4 Exemplary realizations in tools

In the SAFE project the SAFE meta model and the SEP was implemented in Enterprise Architect by using UML and BPMN.

Based on common practices in the automotive domain, safety plan is usually generated by the usage of rich-text capable document processing tools or spreadsheet tools.

6.3.5 Exemplarily usage in industrial use case

The setup of a safety plan was not part of the SAFE use cases.

6.4 Hazard Analysis & Risk Assessment (HRA)

6.4.1 Activities

In oder to determine the risk and define safety goals for the development of a safety related item, a hazard and risk analysis has to be performed. Hazards and hazardous events have to be described such that the safety goals – identified in a further step – can be traced against the hazards. Each hazardous event it shall be clarified

- how the environment might contribute to it
- how the the driver might contribute to it
- how other traffic participants (pedestrians or other vehicles) might contribute to it
- and how the vehicle is contributing to it

Although the ISO recommends that “Hazards shall be defined in terms of the conditions or behaviour that can be observed at the vehicle level”, it is recommended that – for traceability reasons – also the role of the item in a given hazardous event is made explicit. To structure the description of the hazardous events according the above contributing factors, a set text patterns is proposed to support the textual description of hazardous events according to the above contributing factors. Such structured recording of hazardous events allow to better identifiy situations that have not been considered in the hazard identification and therefore supports the goal of this step to to identify all hazardous situations that the item may be involved in.

Safety goals are considered as high level safety requirements and their basic objective is to prevent potential malfunctions from the item to become a hazard on vehicle level.

Note, that SAFE enforces no particular order to carry out design steps and steps of the safety activities.

6.4.2 Formalism in Meta-model

The for factors making up the hazardous events are anchored as follows in the meta-model.

Note the first the three factors (environment, driver, other traffic participants) are “accessible” via the Operational Situation. For details see D3.5.b..

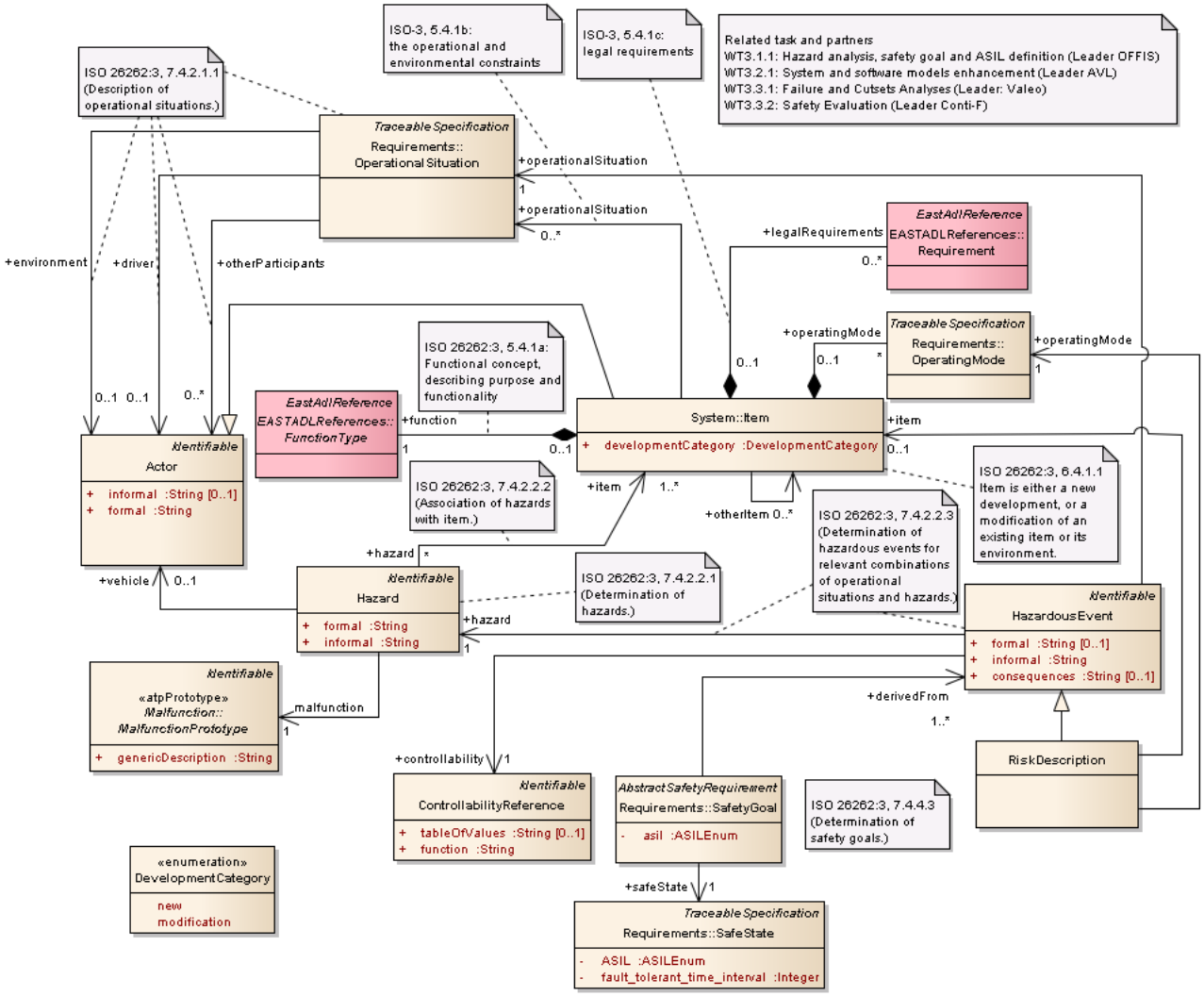

Figure 33: Hazardous events anchored in SAFE meta model

6.4.3 Formalism in SEP

The following activities are modelled in a EnterpriseArchitekt model to represent “Hazard Analysis & Risk Assessment (HRA)” topics.

“[3-7] Hazard Analysis and Risk Assessment” (see chapter 13) in the swimlane “analysis engineering”

“[3-8.4b] Safety Analysis on Item Feature Level” (see chapter 13) in the swimlane “analysis engineering”

“[4-7.4c] Verfication of System Design and Architecture” (see chapter 13) in the swimlane “analysis engineering”

“[6-7x] Execution of software architecture analysis” (see chapter 13) in the swimlane “analysis engineering”

“[5-7x] Execution of hardware safety analysis” (see chapter 13) in the swimlane “analysis engineering”

7 Guidelines and activities in the ISO 26262 development Phase (System)

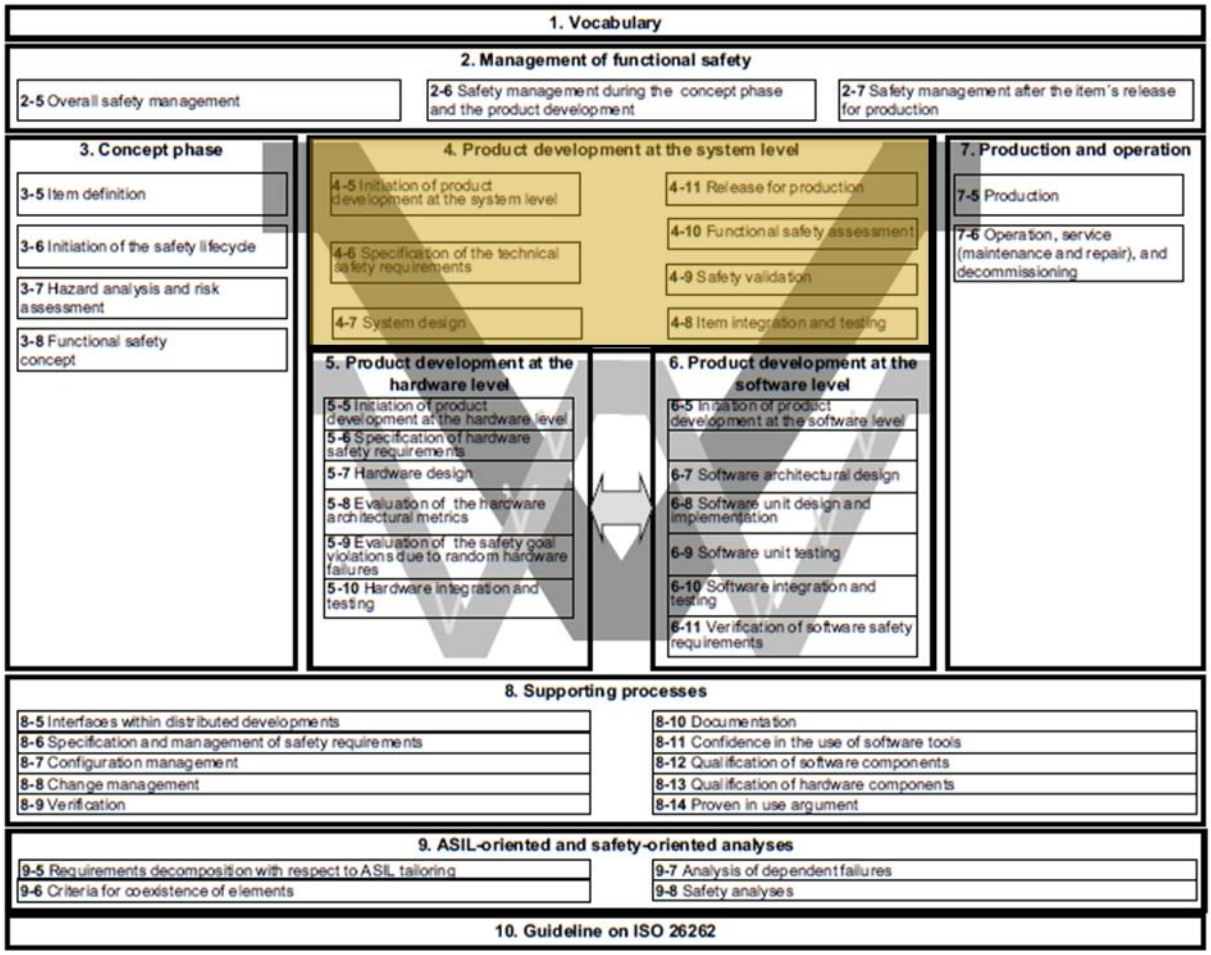

Figure 34: ISO 26262 Product Development at the System Phase

7.1 Functional Safety Concept

7.1.1 Activities

The functional safety concept shall be used to derive the functional safety requirements, from the safety goals, and to allocate them to the preliminary architectural elements of the item, or to external measures.

As described in ISO 26262 part 3 chapter 8 the functional safety concept shall address:

- fault detection and failure mitigation
- transitioning to a safe state
- fault tolerance mechanisms, where a fault does not lead directly to the violation of the safety goal(s) and which maintains the item in a safe state (with or without degradation)
- fault detection and driver warning in order to reduce the risk exposure time to an acceptable interval (e.g. engine malfunction indicator lamp, ABS fault warning lamp)
- and arbitration logic to select the most appropriate control request from multiple requests generated simultaneously by different functions.

7.1.2 Formalism in Meta-model

The Functional Safety Concept shall be used as an add-on to the already existing AnalysisFunctionType of EAST-ADL or any other functional description of the item. It is realized in the Meta-Model by the artifact FunctionalSafetyExtension. This extension shall contain the functional safety requirements that are specified to fulfill the safety goals resulted by the Hazard Analysis and Risk Assessment (see chapter 6.4).

Functional Safety Requirements are realized in the Meta-Model by the artifact FunctionalSafetyRequirement.

The artifact SafetyConcept (part of package “Requirements”) shall be used as container for functional safety requirements that specify the logical elements of the safety architecture/design defined in the Functional Safety Concept.

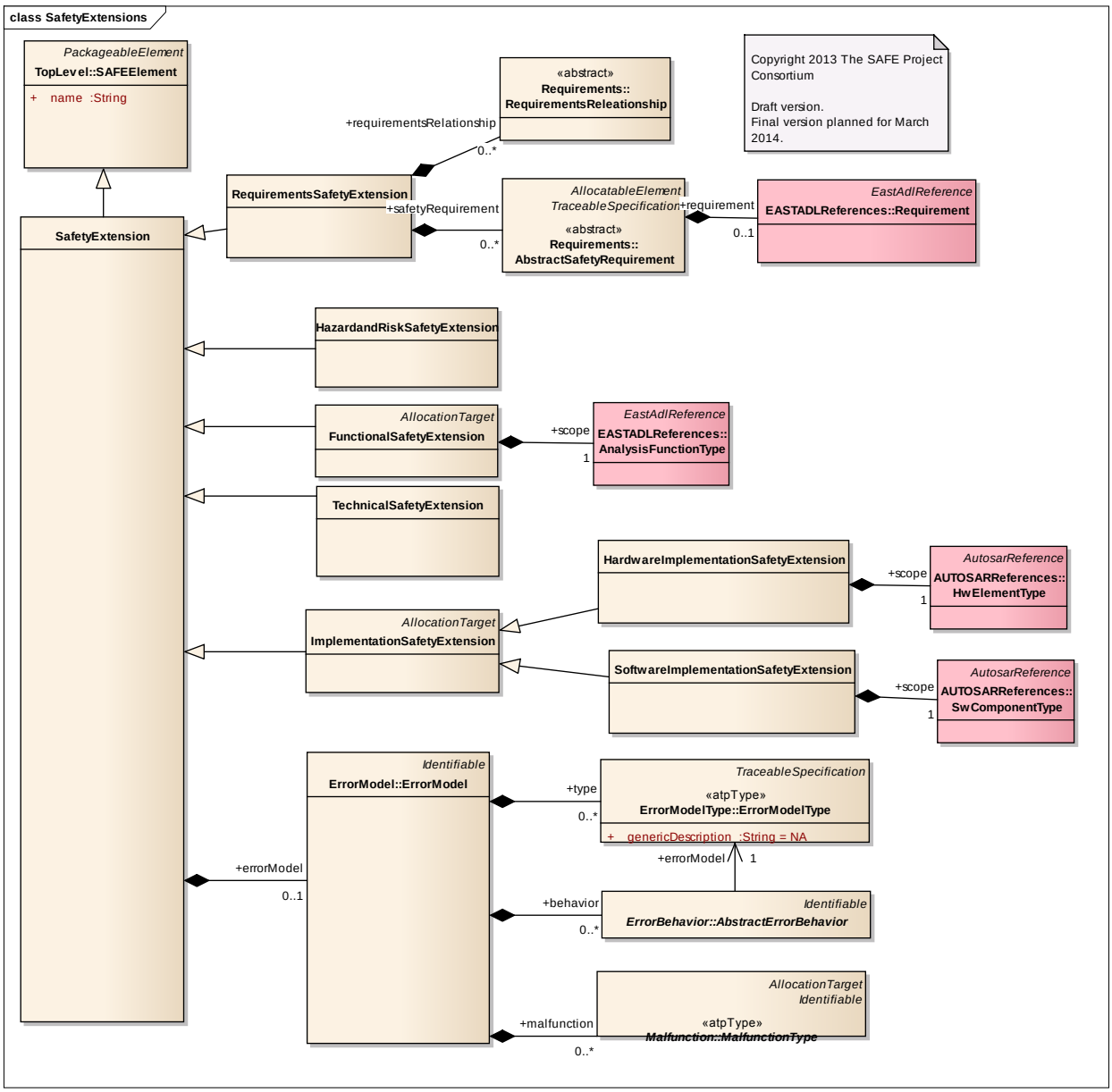

Figure 35: Meta Model – Functional Safety Extension

The following figure is showing the main content of the Functional Safety Concept and its correlations to

- Safety Analyses on System Level

- already existing Archictecture (A)
- already existing Requirements (R)
- already existing Design (D)

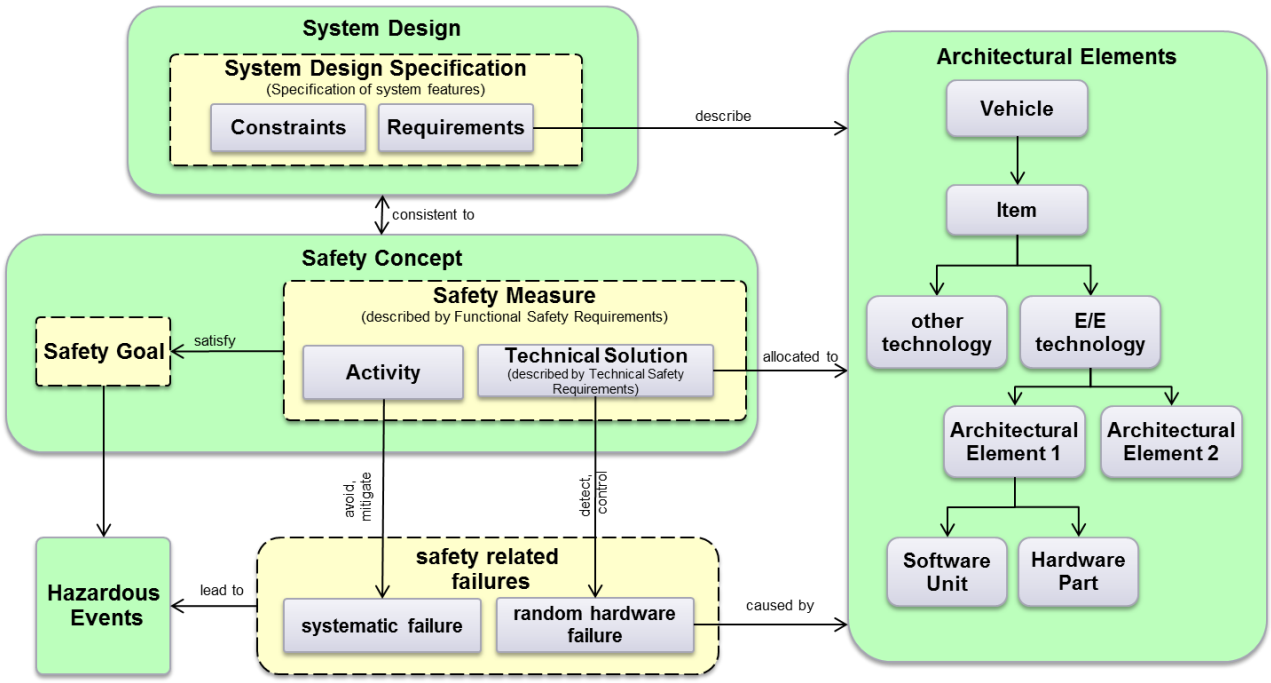

Figure 36: Functional Safety Concept

Further details according to Functional Safety Concept see D3.2.1.c – Functional Safety Concept. and D3.1.2.c - safety requirement expression modeling.

7.1.3 Formalism in SEP

The following activities are modelled in a EnterpriseArchitekt model to represent “Functional Safety Concept” topics.

“[3-7e] Definition of Safety goal and Safe state” (see chapter 13) in the swimlane “requirements engineering”

“[3-8] Definition of Functional Safety Requirements” (see chapter 13) in the swimlane “requirements engineering”

7.1.4 Exemplary realizations in tools

Based on common practices in the automotive domain, Functional Safety Concept is usually generated by rich-text capable document processing tools or spreadsheet tools.

7.1.5 Exemplarily usage in industrial use case

The method described in chapter 5.2 was exemplarily evaluated on a brake system use case. During creation of functional safety concept the systematic top-down analysis of item level malfuncitons was applied as required in the ISO 26262.

7.2 Technical Safety Requirements and System Design

7.2.1 Activities

Technical safety requirements refine the functional safety requirements specified in the functional safety concept. The functional concept and the preliminary architectural assumptions can be considered during specification of technical safety requirements.

Evidence for compliance of technical safety requirements and functional safety requirements shall be provided by using inductive safety analyses e.g. FMEA as method (further information see chapter 5.2.2).

System design and the technical safety concept shall comply with the functional requirements and the technical safety requirements specification of the item.

Verification of system design and the technical safety in SAFE project is partly covered by the safety analyses methods described in chapter 5.2.

If the Technical Safety Concept makes use of Partitioning in order to implement Freedom from interference of SW-Components according to ISO 26262 -6-7.4.11 the partitioning scheme and the partitioning framework of the operating system shall be specified as indicated in Figure 38: ISO 26262 - System Design.

7.2.2 Formalism in Meta-model

The Technical Safety Concept in the model based development is a further maturity level of the architecture. It shall specifiy the technical elements used to realize the logical elements described in the Functional Safety Concept (see chapter 7.1). That process could run iteratively.

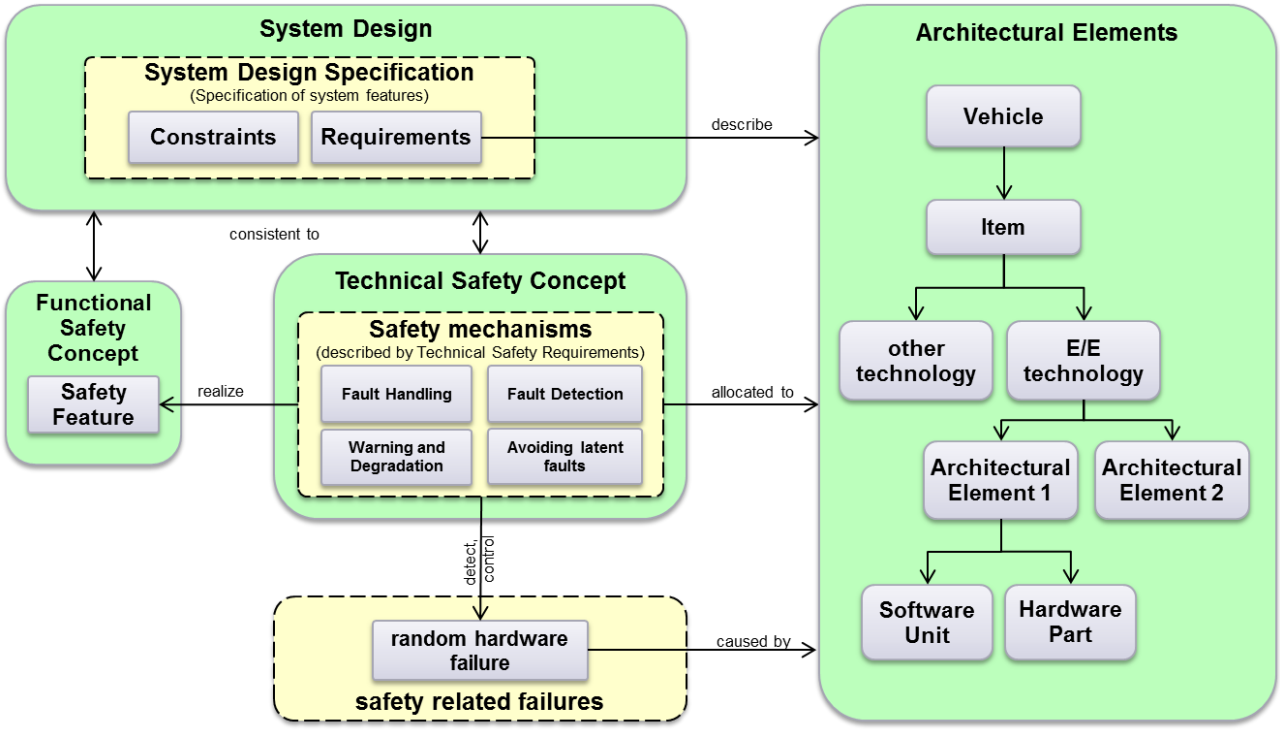

Figure 37: Technical Safety Concept

Further details see D3.2.1.c – Technical Safety Concept

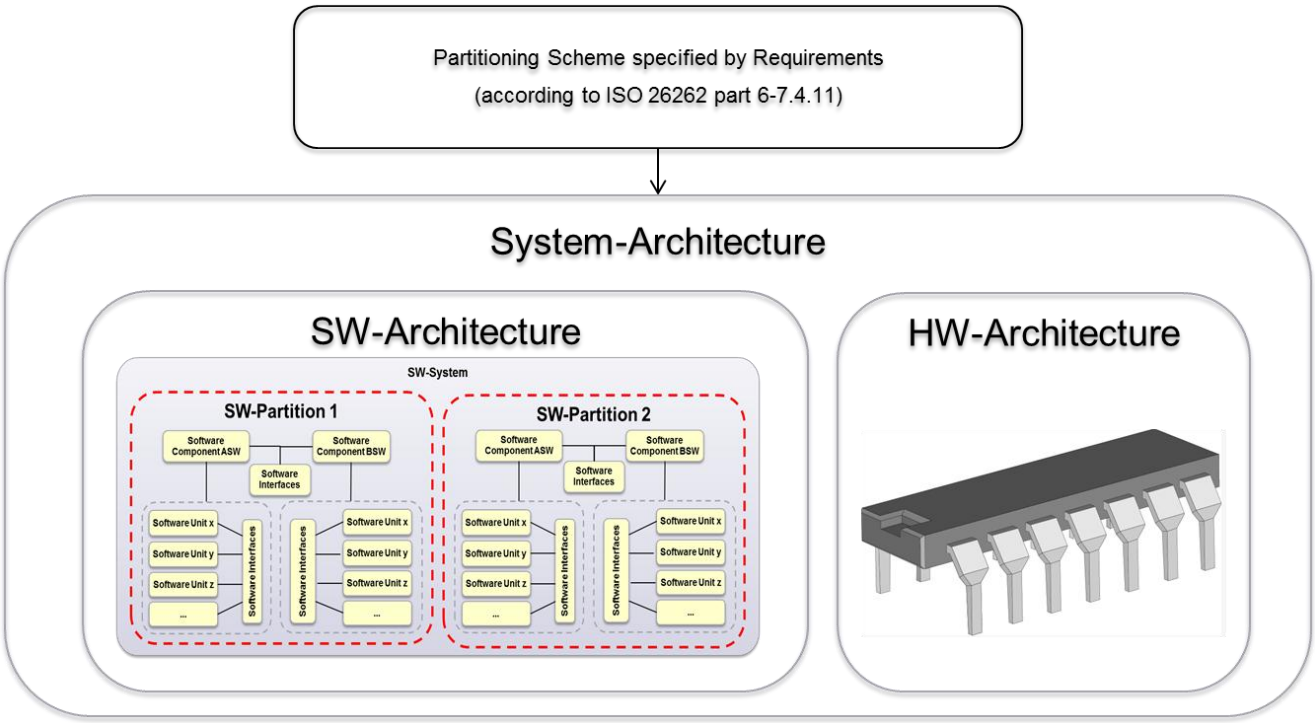

Figure 38: ISO 26262 - System Design

7.2.3 Formalism in SEP

The following activities are modelled in a EnterpriseArchitekt model to represent “Technical Safety Concept” topics.

“[4-6] Definition of Technical Safety Requirement” (see chapter 13) in the swimlane “requirements engineering”

“[3-8.4a] Allocation of FSR to ItemArchitecture” (see chapter 13) in the swimlane “architecture engineering”

“[4-7.4a] Definition of System Architecture” (see chapter 13) in the swimlane “architecture engineering”

“[3-5b] Definition of a preliminary item architecture” (see chapter 13) in the swimlane “design engineering”

“[3-8.4c] Definition of Safety Feature-Design on Item Level” (see chapter 13) in the swimlane “design engineering”

“[4-7] Definition of System Design” (see chapter 13) in the swimlane “design engineering”

“[4.7.b] Definition of safety-relevant HW-SW Interfaces” (see chapter 13) in the swimlane “design engineering”

8 Guidelines and activities in the ISO 26262 development Phase (Software)

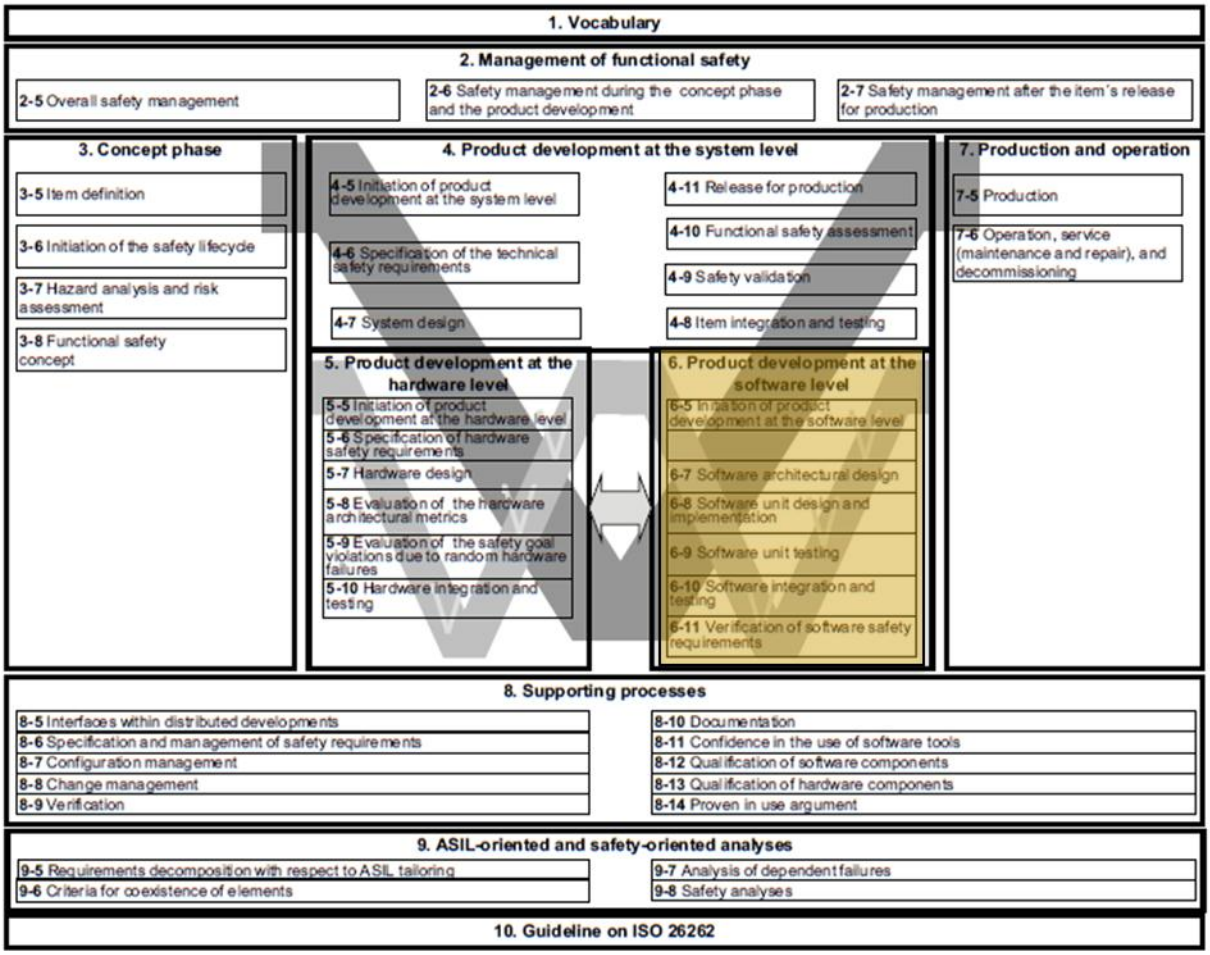

Figure 39: ISO 26262 Product Development at the Software Level

8.1 Software Safety Requirement Specification

8.1.1 Activities

The software safety requirements specification is the first objective of this sub-phase. They are derived from the technical safety concept and the system design specification.

Activities to detail hardware-software interface requirements are the second objective initiated in ISO 26262-4:2011, Clause 7.

To verify the software safety requirements and the hardware-software interface requirements are consistent with the technical safety concept and the system design specification is the third objective.

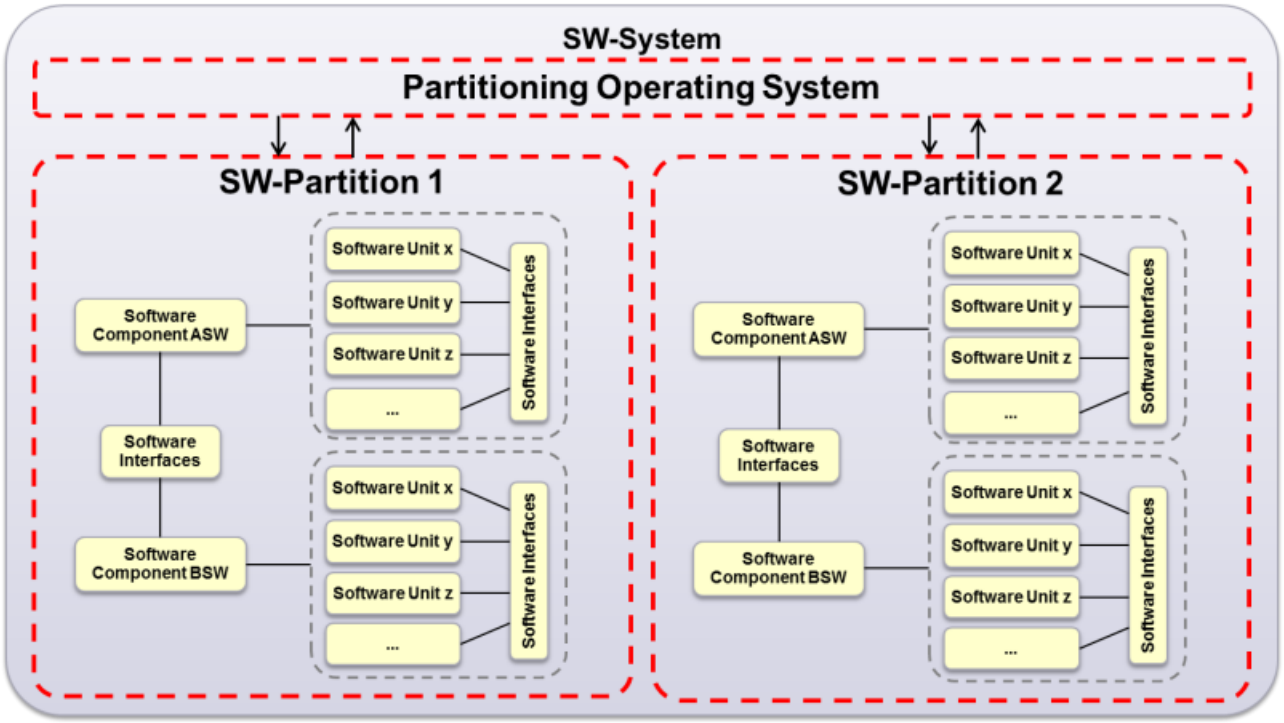

Figure 40: SW System Architecture

8.1.2 Formalism in Meta-model

Software Safety Requirements are realized in the Meta-Model by the artifact SoftwareSafetyRequirement.

The artifact SafetyConcept (part of package “Requirements”) shall be used as container for software safety requirements that specify the safety mechanisms realized by software.

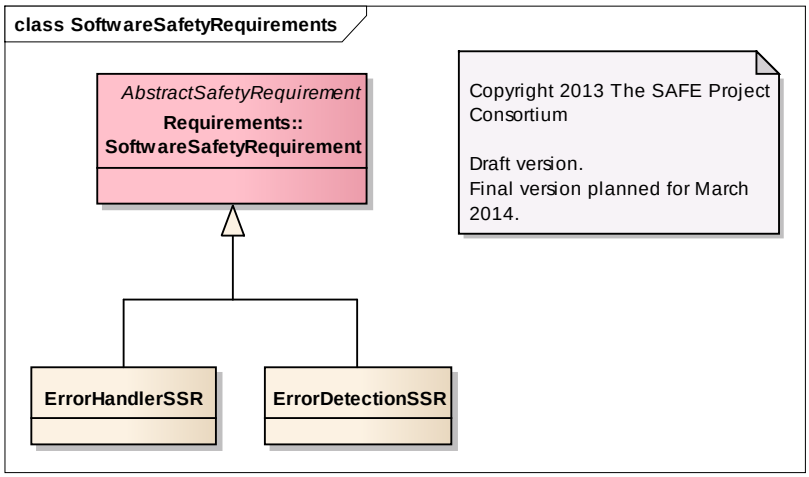

Figure 41: Meta-Model - SoftwareSafetyRequirements

Further Details according to Software Safety Requirement Specification see D3.2.1.c – Software Level.

8.1.3 Formalism in SEP

The following activities are modelled in a EnterpriseArchitekt model to represent “Software Safety Concept” topics.

"[6-6] Definition of Software Safety Requirement" (see chapter 13) in the swimlane "requirements engineering"

"[6-8] Definition of software unit requirements" (see chapter 13) in the swimlane "requirements engineering"

8.1.4 Exemplary realizations in tools

Based on common practices in the automotive domain, Software Safety Requirements are usually generated by rich-text capable document processing tools or spreadsheet tools.

8.1.5 Exemplarily usage in industrial use case

The evaluation scenario "Safety Code Generation" applies the concept of software safety requirement specification according to SAFE. This is done by application the SAFE tool platform incl. the tree editor for modeling safety requirements. As there is a close relation between the evaluation scenario and the method definition, the SAFE modeling approach for safety requirements fits very well to the evaluation scenario, allowing to generate software safety mechanisms from the formalized requirements specification.

8.2 Software Realization

8.2.1 Activities

The software assets of the item under development are realized based on software safety requirements. The software safety requirements are the first artifacts to be produced within the product development at the software level as defined by the ISO 26262. These requirements are derived from the technical safety requirements and are based on the technical safety concept, the system design specification and the hardware software interface specification.

After being specified, software safety requirements must be realized. One could use manual coding and a common programming language such as C++, a modeling language such as Simulink® or a domain specific language (DSL) together with model transformations for realizing software safety requirements.

The ISO 26262 recommends the use of clear semantics, strong typing, low complexity, small component size, amongst others, for the realization of software assets. Manual implementation guaranteeing these characteristics leads to increased verification efforts since the manual steps involved in the processes are error-prone. Therefore, an automated approach for the realization of software safety requirements is beneficial.

In order to automate the realization of software safety requirements, the identification of recurring patterns in the specification of these requirements was realized and the patterns in turn formalized using a meta-model. A DSL is used to provide predefined constructs for the specification of software safety requirements according to the meta-model and thus making it easier to specify such requirements. Furthermore, it is possible to seamlessly integrate the software safety requirements to existing development processes, tools and methodologies.

Using model transformation frameworks the meta-model elements can be transformed into code and models (e.g. C-code and AUTOSAR XML). The verification effort is shifted from the manual coding towards the model transformation or to visual inspections of the generated artifacts. The realization of software assets through generative approaches reduces the chances of human-induced errors. Furthermore, it provides a formalized link between the technical safety

requirements to the software safety requirements and a link between the software safety requirements and the generated software assets as required by the ISO 26262.

The specification and realization of software safety requirements done according to the meta-model can be used as argument for the safety case documentation, for the verification of compliance to the technical safety requirements (ISO 26262-6 6.4.8) and for documenting ASIL rational according to the ISO 26262-9 Clause 5. Moreover, formally defining software safety requirements provides support for formally specifying and automatically implementing some classes of test cases.

8.2.2 Formalism in Meta-model

SAFE uses the standard AUTOSAR templates (e.g. Software Component Template) to describe the software architecture. Regarding the specification of software safety requirements, the SAFE Meta-model has been adapted at several points.

Tactics

The concept of “Tactics” has been introduced, which describes the semantics of the software safety requirement. E.g. a software safety requirement X has the tactic “Error detection” and detects several kinds of errors. Another software safety requirement Y has the tactic “Error handling” and is able to work with previously detected errors.

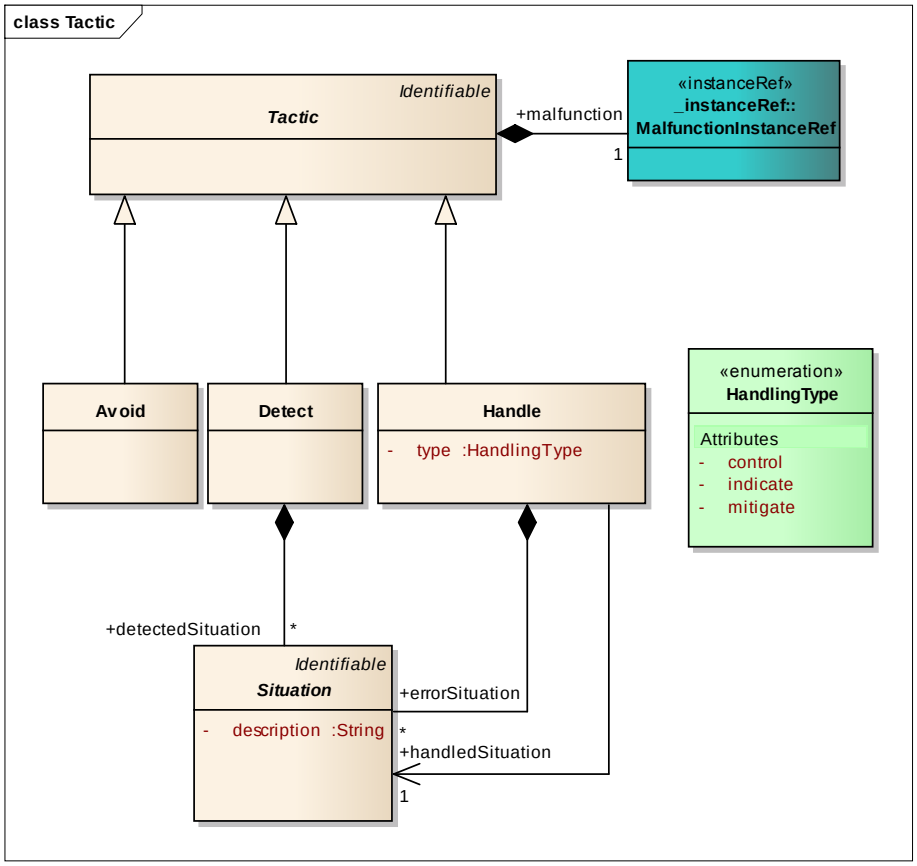

Figure 42: SAFE meta model malfunction instance

Requirements Specification

In addition, a formalism to describe software safety requirements has been introduced to model the required attributes for those requirements.

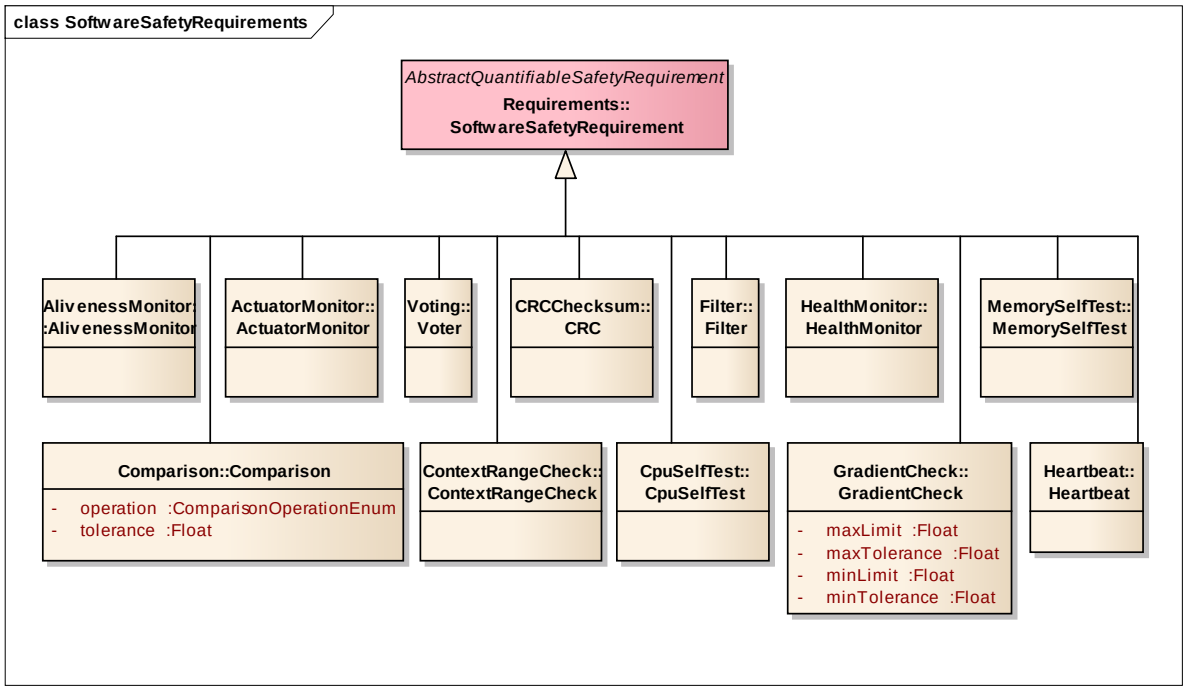

Figure 43: SAFE meta model SoftwareSafety Requirements

Concrete software safety requirements configuration

In addition, the concrete software safety requirements have been further detailed, to precisely describe the required information. In the following, we show the Meta-model for the gradient checker in more detailed. For a complete overview of the several software safety requirements, please refer to the main deliverables of SAFE WT 3.6 “Safety Code Generation” [1].

8.2.3 Formalism in SEP

The following activities are modelled in a EnterpriseArchitekt model to represent “Software Realization” topics.

“[6-7x] Definition of software architecture” (see chapter 13) in the swimlane “architecture engineering”

“[6-8x] Definition of software design” (see chapter 13) in the swimlane “design engineering”

9 Assessment activity / architecture model for functional safety development (AAM)

9.1 Introduction - General description of assessment activity/architecture model for functional safety development (AAM)

The automotive specific functional safety norm ISO26262 [4] defines process requirements for functional safety-aware development in the automotive domain. It has high demands on process documentation and analysis. Some of the system characteristics important in the context of the ISO26262 are also relevant for non-safety related development and are therefore already addressed in conventional models. However, it is currently not clear how the development view and models necessary for safety documentation and analysis can and should be integrated in order to minimize modeling effort, to keep consistency between artifacts and to enable effective reusability and change management. Methods which allow demonstration of functional safety of automotive products according to ISO26262 are needed to be applicable to such an integrated model. While AUTOSAR [2] provides some technical prerequisites necessary to realize safety relevant systems, such as protection mechanisms or safe end-to-end communication, it is not yet clear how to use the AUTOSAR methodology within an ISO26262 compliant process.

The above challenges must be addressed if the European automotive industry is to cope with the increasing vehicle system complexity and a massive increase in safety-relevant functions (e.g. for driver assistance systems or electrical or hybrid vehicles). They can only be tackled effectively in a joint initiative that includes the complete automotive supply chain (OEMs, Tier 1's, Silicon vendors and tool suppliers) as well as academia that provide a significant research background in relevant fields. The European funding project SAFE addresses these challenges and speeds up the efficient development of safety critical features in cars. The objective is to enhance method, e.g. for defining safety goals and define development processes compliant with the ISO26262 standard for functional safety in automotive electrical and electronic systems.

The project started July 2011 and published the concepts, an integrated meta-model and an Eclipse based open source technology platform in early 2013. This document is a starting point for the process and assessment model [6].

Target of this document is a reference process model for functional safety assessment activities based on required functional safety activities according ISO 26262 and the description of the methodology. The methodology is based on results from the concepts (developed in WP3) and should deliver templates or guidelines to apply automated model-based verifications (in the meaning of ISO 26262).

The AAM is closely related to the result of the guideline (see description of task 3 above) and the collected methods linked in this guideline. The analysis of dependent failure is taken as an input for identification of the structure for AAM. The AAM provides at the end further content to the guideline.

Attached picture should show the dependency between other related projects and work-tasks within the SAFE project.

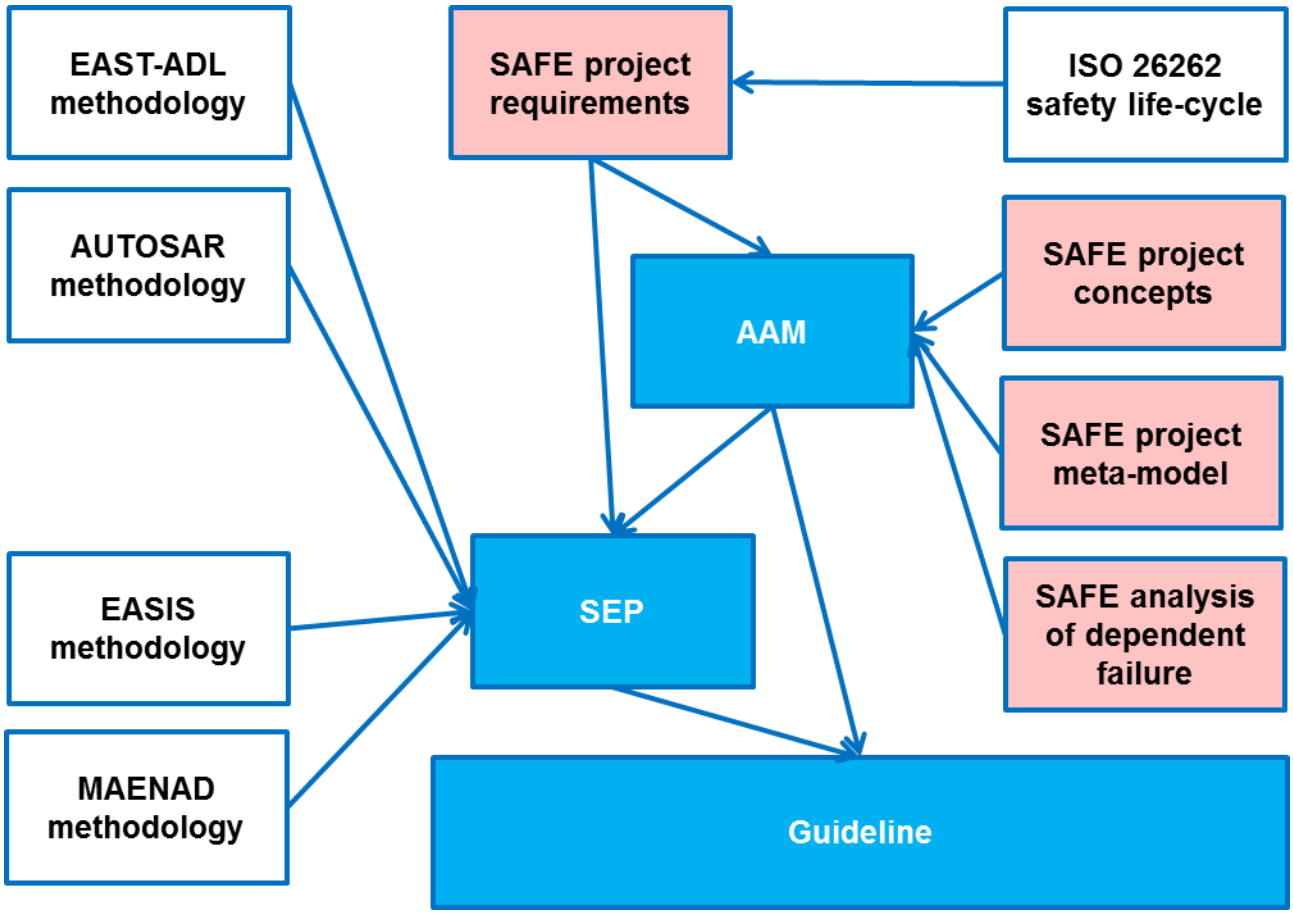

Figure 44: relationship between introduced tasks. White fields list material from external sources.

9.2 Description of activities

9.2.1 ISO26262 as the starting point

ISO 26262 introduces 3 different confirmation measures:

- Functional Safety Audits
- Confirmation Reviews
- Functional Safety Assessment.

The means of those measures are given in the following table from ISO 26262, Part2, Table 2.

Table 2 — Procedural requirements for confirmation measures

Topic	Confirmation review	Functional safety audit	Functional safety assessment
Subject for evaluation	Work product	Implementation of the processes required for functional safety	Item as described in the item definition in accordance with ISO 26262-3:2011, Clause 5
Result	Confirmation review report[a]	Functional safety audit report[a] in accordance with 6.4.8	Functional safety assessment report in accordance with 6.4.9
Responsibility of the persons that perform the confirmation measure	Evaluation of the compliance of the work product with the corresponding requirements of ISO 26262	Evaluation of the implementation of the required processes	Evaluation of the achieved functional safety Provision of a recommendation for acceptance, a conditional acceptance or a rejection, in accordance with 6.4.9.6
Timing during the safety lifecycle	After completion of the corresponding safety activity Completion before the release for production	During the implementation of the required processes	Progressively during development, or in a single block Completion before the release for production
Scope and depth	In accordance with the safety plan	Implementation of the processes against the definitions of the activities referenced or specified in the safety plan	The work products required by the safety plan, the implementation of the required processes and a review of the implemented safety measures that can be assessed during the item development

[a] This report can be included in a functional safety assessment report.

Figure 45: Requirements for measures

9.2.2 Model-based Development and Simulations

The development of products with support by model based engineering is already addressed in ISO 26262. Since ISO 26262 does not address any process iterations, it is a matter of interpretation to assure the specific requirements from ISO 26262. The iteration is addressed e.g. in the verification of architecture:

Table 3 — System design verification

Methods		ASIL			
		A	B	C	D
1a	System design inspection[a]	+	++	++	++
1b	System design walkthrough[a]	++	+	o	o
2a	Simulation[b]	+	+	++	++
2b	System prototyping and vehicle tests[b]	+	+	++	++
3	System design analyses[c]	see Table 1			

[a] Methods 1a and 1b serve as a check of complete and correct implementation of the technical safety requirements.

[b] Methods 2a and 2b can be used advantageously as a fault injection technique.

[c] For conducting safety analyses, see ISO 26262-9:2011, Clause 8.

Figure 46: System design verification

Table 6 — Methods for the verification of the software architectural design

	Methods	ASIL A	ASIL B	ASIL C	ASIL D
1a	Walk-through of the design[a]	++	+	o	o
1b	Inspection of the design[a]	+	++	++	++
1c	Simulation of dynamic parts of the design[b]	+	+	+	++
1d	Prototype generation	o	o	+	++
1e	Formal verification	o	o	+	+
1f	Control flow analysis[c]	+	+	++	++
1g	Data flow analysis[c]	+	+	++	++

[a] In the case of model-based development these methods can be applied to the model.

[b] Method 1c requires the usage of executable models for the dynamic parts of the software architecture.

[c] Control and data flow analysis may be limited to safety-related components and their interfaces.

Figure 47: Methods for verification

It is up to the tailoring of the lifecycle if the focus is more on simulations or on prototyping. Simulations are generally seen as a method for verification. In model-based development, it is a basic requirement to verify the correctness of the model used for the simulation, before the model could be used to verify the prototype or the realized product or characteristics, behavior or parts of it.

9.2.3 Hierarchical Error Analysis

Functional modeling and safety analysis [5] are two important aspects of safety-critical embedded systems. However, they are often conducted separately. Following SPES2020 [7] both aspects are called perspectives. The following figure introduces further perspectives, but we will concentrate on the functional perspective and the ISO26262 perspective. Each cell in the matrix of abstraction levels versus perspectives provides a view.

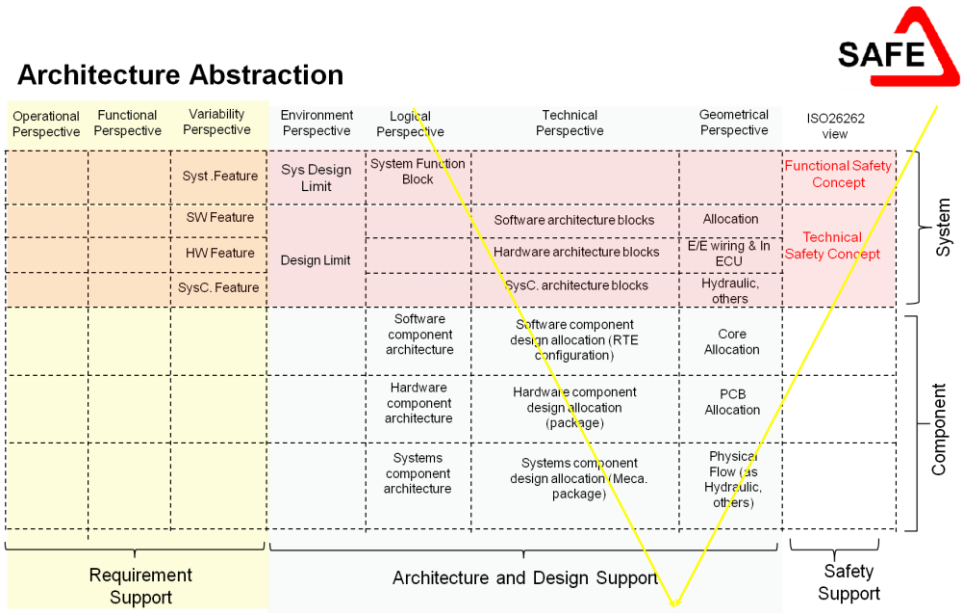

Figure 48: Abstraction levels versus Perspectives

A system model that takes failures into consideration is crucial in ensuring that safety is considered throughout the development process. It will offer the following benefits.

The model will allow engineers to be knowledgeable about the undesirable conditions and system failures and to understand how the behavior of the system is affected by these failures. It will help them to understand the interaction between the software and other system components. The model will also identify the components that are responsible for the system functions that were previously identified by the hazard analysis. These components should then be given special attention in the system development process.
As shown, system, hardware and software modeling is usually done by hierarchical modeling – from an abstract concept to a detailed design. The safety analysis has to be adapted to these abstraction levels, too.

In a safety-critical system, every major failure is classically represented by a fault tree. Each fault tree describes how the individual fault components combine to result in an undesirable system behavior or catastrophic failure. The root of a fault tree represents the major failure or the most general failure. As we go down the tree, the nodes represent more specific or detailed faults. Thus, a fault tree describes the catastrophic event in terms of its causal factors or faults in a hierarchical fashion.

The following figure shows an example of architectural layers corresponding to a fault-tree analysis. The analysis of a fault tree could be done either qualitatively or quantitatively. A model transformation from a fault tree model to another safety analysis model (e.g. reliability block diagram) should be discussed in further studies. A mixed safety modeling strategy will combine advantages of all safety models and reduce their drawbacks.

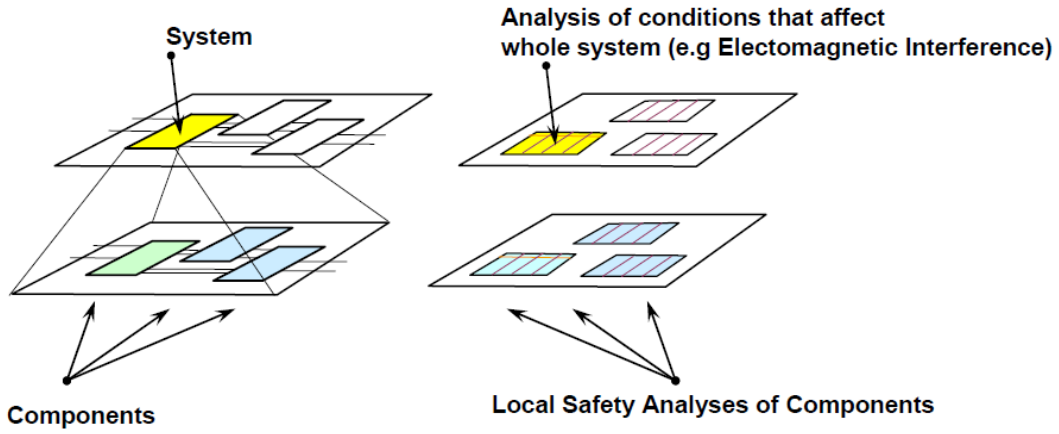

Figure 49: Abstraction between system- and component-layer

In the abstraction levels of EAST-ADL, safety analysis models are applicable. An early safety evaluation will be performed to demonstrate the diagnosis coverage using model based FTA analysis and error hazard occurrence, based on the error models propagation between safety critical and safety relevant functions/features.

Especially, the improvement and the definition of interfaces between requirements, architecture, design and its verification during development via Safety Analysis will be highlighted. The definition of the interfaces, the identification of functional and non-functional requirements, technical requirements, design limitations and preliminary architectural assumptions will be clarified, and assigned to the different hierarchical levels such as vehicle, system or component. A secondary target is to avoid or mitigate systematic failure at the interfaces to supplier and customer.

Insure a seamless handling of safety requirements within overall requirement management providing relevant coverage and impact synthesis for the safety case documentation. Avoid inefficiency of document oriented traceability by introducing model centric requirement management in design activities (refer to dysfunctional modeling improvements).

By merging or at least coupling functional and dysfunctional modeling while sharing common

abstraction levels, consistency of the overall safety concept is achievable with an optimized effort. Furthermore sharing the same ground between designers and safety experts insure consistency during the complete lifecycle and especially while iterating the different increments or during maintenance.

By studying all relevant accidental events (that have been identified by a preliminary hazard analysis, a HAZOP, or some other technique), the ETA can be used to identify all potential accident scenarios and sequences in a complex system.

The purpose of hazard analysis is to examine the system and determine which components of the system may lead to a mishap. There are two basic strategies to such analysis that have been termed inductive and deductive. Essentially, inductive techniques, such as event tree analysis and failure modes and effects analysis, consider a particular fault in some component of the system and then attempt to reason what the consequences of that fault will be.

Deductive techniques, such as fault tree analysis, consider a system failure and then attempt to reason about the system or component states that contribute to the system failure. Thus, the inductive methods are applied to determine what system states are possible and the deductive methods are applied to determine how a given state can occur.

Due to the sound basis of functional / dysfunctional modeling it will be possible to capture elements and feed inputs in FMEA and FTA thus avoiding double filling and synchronization issues between design and safety teams. Dysfunctional modeling will allow some automatic computation in the safety analysis allowing safety experts to focus on critical topics. Above improvement on the coupling with design, lowering the effort to critical issues shall also allow to be more reactive during increments.

The Fault Tree Analysis (FTA) [3] is a top–down method that systematically breaks down hazards to their causes. The result is then visualized in a tree structure. Thus, the FTA rather helps systematically analyzing hazards than to detect them.

The first step in the FTA is to define the system bounds. This includes the definition of the hazard, the events the analysis has to take into account, the physical system bounds, and the initial system state. For example, there is a huge difference between a car in city traffic and a car on the autobahn being analyzed. Once the system bounds are defined, the occurrence of a hazard (top–event in an FTA) is iteratively decomposed into its causes. To visualize this process a standardized graphical notation (IEC 1025 standard) is used.

The Failure Mode and Effects Analysis (FMEA) [3] represent a preventive safety analysis approach. It is utilized to identify and assess potential failure causes as early as possible, e. g. during the early design phases. This helps to prevent control and further failure costs during the production or even the operation phase. Furthermore, a systematic failure analysis approach inhibits further repetition of design faults in other products.

FMEAs should be applied in early product life cycle phase's, e.g. the concept and design phase, since it proved to be most beneficial for a cost–benefit analysis. The sooner a fault is discovered, the cheaper its correction can be.

A reliability block diagram (RBD) is a diagrammatic method for showing how component reliability contributes to the success or failure of a complex system. RBD is also known as a dependence diagram (DD).

A RBD or DD is drawn as a series of blocks connected in parallel or series configuration. Each block represents a component of the system with a failure rate. Parallel paths are redundant, meaning that all of the parallel paths must fail for the parallel network to fail. By contrast, any failure along a series path causes the entire series path to fail.

An event tree analysis (ETA) is an inductive procedure that shows all possible outcomes resulting from an accidental (initiating) event, taking into account whether installed safety barriers are functioning or not, and additional events and factors.

Final objective of the whole set of analysis methods is to allow continuous verification while walking through the development cycle and involving the different development teams.

This leads to a set of qualitative and quantitative measures:

- Safety concept consistency insured through relevant abstraction levels
- Efficient modeling mixing functional and dysfunctional focuses
- Formal exchange with OEM and subcontractor organizations based on models
- Consistency of safety analyses done in the different levels (hierarchical links, impacts)
- Efficiency of automated safety analyses realization and maintenance
- Consistency of safety traceability with overall traceability
- Efficiency of model centric requirement management
- Efficiency of safety products developments by tight coupling of designers with safety experts sharing the same technical ground

9.2.4 Verifications by Safety Analysis

Safety analysis methods are basically just special methods for verification. Particularly the different FMEA methods support the verification of systems.

A System-FMEA primarily supports the verification of requirements and their allocation to functions as well as to logical or technical elements. A Design-FMEA questions the correct interpretation of the design or implementation, due to the criticality of the failure effect a risk based approach for verification measure could be provided. This is usually started with the design concepts in the later iterations it incorporated to the realized product. The Design-FMEA primarily supports the design verification and is finalized by a Design Review of a cross-functional team. This has also strongly points to the so-called Toyota-FMEA (DRBFM -Design Review Based on Failure Modes). Usually with a Process-FMEA the production process should be analyzed. Formally it would be possible to any process to be analyzed by this method, see also the chapter "Process Verification." In any FMEA standard requires a final review to confirm the goal achievement of the analysis. A final review of the FMEA is formally part of any FMEA method.

The following verifications can be supported by safety analysis:

Completeness of the relevant safety goals

Primarily safety goals are as follows: "Avoid that a possible malfunction of the item could possibly cause harm." Any malfunction can be structured in a System-FMEA as effects of systems failure. Any credible effect of a systems error could be considered as a malfunction that violates a safety goal. If all potential systems error or failure and their effects are considered and no effects lead to any other safety relevant effect (top-failure) than the defined Safety Goal, the completeness could be demonstrated.

Completeness of relevant functions within the boundary of the item

This analysis is based on the functional networks of the VDA FMEA. However, automated testing would be much more effective by using architecture tools. Checking may take place in any horizontal level of abstraction. Since a System-FMEA could be performed on any level of abstraction any completeness of functions within a element boundary could be analyzed on software-, hardware- component level and even within silicon, such as semiconductor. It is comparable with branch checks in SW-units, it analysis on a similar way that inputs and outputs within a boundary are complete connected. The basic principle of the analysis is to identify the signal chain, which was developed by Robert Lusser and had been described over 80 years ago.

Consistence and completeness of dedicated functions from a higher level derived to a lower level of horizontal abstraction. (Verification of function decomposition)

It is comparable with the analysis of completeness of functions within a boundary in previous chapter. This analysis does extend the analysis and compares the already approved on a higher

level of horizontal abstraction with the same representation of the function on a lower level. Depending on the criterions which were added in the lower level of abstraction, their completeness could be also evaluated. It could be based on the function network of a VDA FMEA, but as well as in the previous analysis; better transparence could be achieved by architecture tools. A signal chain on system level could be compared with a signal chain which is allocated to software or hardware. In combination with previous analysis also the hardware-software-interface could be analyzed, due to separation of an element on higher level into 2 or more elements on lower level of abstraction. Further abstractions within the same level of abstractions could be analyzed for completeness and correctness, by adding information about environmental impacts, power supply, voltage, EMC, common usage of resources. Due to those verifications the analysis of dependent failure could be supported as described in chapter 5.1.5.

Consistency check of the interfaces (Verification of product decomposition)

The VDA FMEA by the structure networks, the interfaces for the entire product structure are described. Here there is the challenge that functional and technical interfaces are not always congruent. By comparing functional, logical and technical structure between each other and between structure and between their interfaces in different horizontal level of abstractions could provide information about completeness and consistency of those structures and their interfaces. Also here architecture tools and possible routines are much more effective than static structure within a VDA FMEA.

Completeness of the considered malfunctions (failure, error or fault modes)

Especially during deductive analysis, it is important to argue a certain completeness of considered malfunctions. Basically any characteristic of a function or an element could fail and so having potential impact to malfunctions. Any identified error of goods could be considered as an argument to add measure to improve measures during development and for implementing in the product to improve non-functional requirements such as safety, availability or reliability. Since we consider that system elements are always have to correct interact to perform a required function, error modes per functions could be defined. One way could be to apply law from DeMorgan, it converted negated “or” in “and” gates. A VDA FMEA failure analysis, which is seen as the third step of that FMEA approach after product and function decomposition you determine for any function independent from the level of abstraction possible malfunctions. For the verification of the safety requirements, it is first of all necessary to determine completeness related to the allocated function. That means any required characteristics and any required technical behavior and their characteristics could deviate from their intended or required state. By pure information completeness of considered malfunction could be achieved that any information could be wrong. It is recommended by automated checker to consider in addition to that that the information could not be available at the required point. These could provide a completeness argument for the considered malfunctions. In a more deep analysis the following malfunctions could be considered:

- no function
- unexpected function (crosstalk from other systems)
- systematically falsified information or function (for example, signal drift)
- sporadically or improper function or unexpected information
- module or element was not executed addressed or considered
- function or element does not run continuously or is not considered continuous (uninterrupted operation is not, oscillations)
- Wrong Timing

These questions are the basics for the most deductive methods such as HAZOP and Fault Tree Analysis. In essence, they are comparable with the tables in Part 5, Annex D of ISO 26262, which are the basis for the diagnostic coverage. Even in Design-FMEA such analysis is considered to be evaluated sufficiently or necessary coverage of adequate design assurance measures.

Completeness of the considered single point malfunctions (failure, error, faults)

This is the classic domain of FMEA; here all possible malfunctions of an appropriate level to consider whether they can propagate to higher level up-to a safety goal.

Complete view of error combination up-to the order of 2 (e.g. double faults)

Multiple point errors always make high permutations based on their factors, therefore, even in a simple system the analysis of multiple faults sis a challenge. By considering safety mechanisms as a barrier preventing errors from propagation, any fault could be considered as a single point fault related to the barrier or safety mechanism. For the safety goals higher than ASIL C, also fault combinations have to be controlled, depending on their probability of occurrence at the same time. If a safety mechanism is an independent measure to the dedicated safety related function, an error of the safety mechanism could not lead to a failure of the safety related function, so that the these errors could be considered as a double fault. As a consequence any secondary independent function, that could not influence a safety goal by itself, have at least a distance of 2 related to their fault propagation, it means it is at least a double fault related to the considered safety goal. Due to classifying functions into secondary independent functions related to the safety goal, their errors could be considered as double failure.

Correctness of the safety goal itself

In case of considering completeness of hazardous events, the propagation of potential malfunctions of system or item to those hazardous events could be analyzed. In case of complete effect of any malfunction to the considered hazardous event it could be used as an argument. In the domain of event-tee-analysis (ETA) even the combination with relevant driving situations could be considered. Completeness could be argued in case of considering completeness of those driving situations.

9.2.5 Safety Validation

Validation is not in the scope of Safe. Therefore a mayor step before the assessment of functional safety has not been considered. It is a strong recommendation for future activities.

9.2.6 Functional Safety Assessment

ISO 26262 precise the requirements for Functional Safety Assessments in Part 4 chapter 10.

The safety case is considered as an input of the Functional Safety Assessment, but the “Functional Safety Assessment Report” is an input for the Safety Case (further details about safety case generation see WP3). It shows that activities should be performed in parallel. After a successful run of a functional safety assessment, ISO 26262 defines the “Release for Series Production” in its chapter 11.

Due to permanent need of human interactions for analysis, verifications, design decisions, validations etc. within “Safe” only partially the “Functional Safety Assessment” could be considered. Some of the described methods for verification give already the hint, that for complete Functional Assessment a complete tailored safety Lifecycle need to be considered, including human influences.

9.2.7 Confirmation Measures based on “Safe”

Basic process element for the tailoring of the necessary activities derived from the ISO 26262 Safety Life Cycle are decribed in the following basic process element (further details see chapter 13)

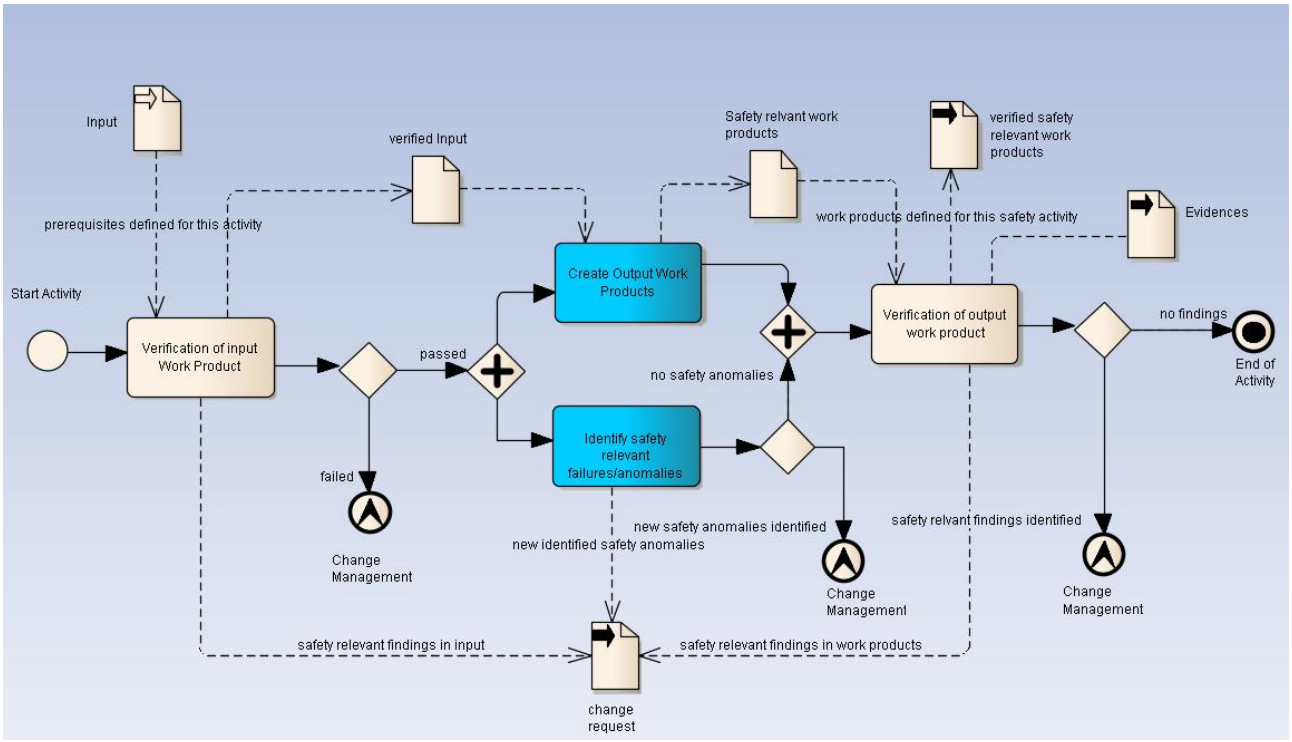

Figure 50: Basic Process Element according "Safe".

All Safety Activities for the development of an Item according ISO 26262 require a basic process structure according the figure above.

The following key characteristics are unique for any activity:

- Product are specified by the following artefacts
  - requirements (R) (e.g. in natural language, according ISO 26262 Part 8 chapter 7)
  - architecture (A) (e.g. structure, behavior)
  - design (D) (e.g. constraints, lists, drawings, 3d)
- Malfunction (including notations for loss of functions) are linked wether to requirements, architecture or design. But they do not belong to the system- or product model.
- Any input of an activity shall be verified for
  - correctness
  - completeness
  - consistency
- Parallel to any safety activity a safety analysis is required, e.g. considering error impacts and propagations.
- The output of any safety activity shall be verified for:
  - correctness
  - completeness
  - consistency
- Deviations or abnormalities from verifications and safety analysis lead to change requests.
- After changing the product specification (based on modified or enhanced R, A or D) the full verification and analysis shall be repeated, by taking credit from previous iteration.

The change shall be in line with the following figure:

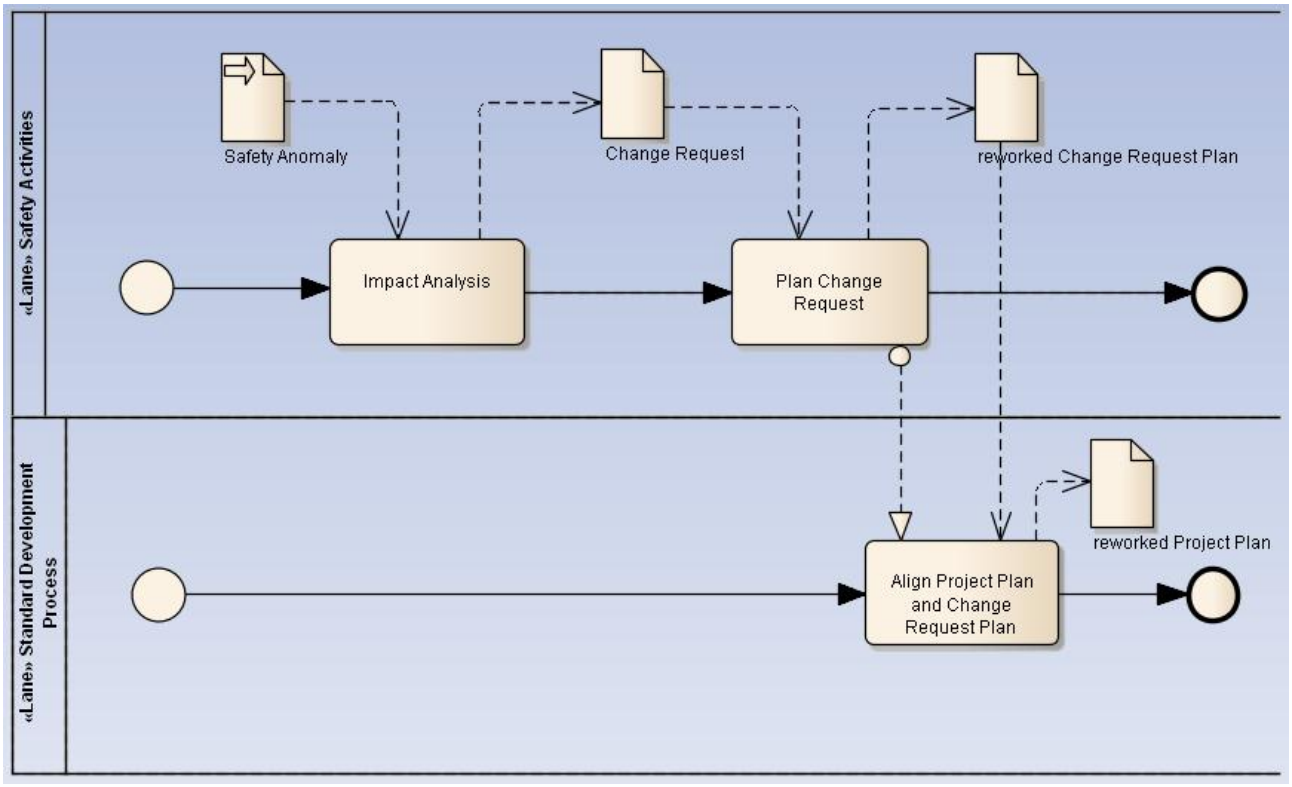

Figure 51: Basic Process Element for changes according "Safe".

Before any change of R, A or D an impact analysis shall ensure limited influence to other product charaterists (entire R, A, and D).

The basic process element could be the basis for the follwojng activities:

- vehicle view -> Activity = Item Definition; Safety Analysis = Hazard & Risk Analysis
- Logical functional view -> Activity = Functional Safety Concept; Safety Analysis = deductive requirement analysis and inductive verification
- Technical functional view -> Activity = Technical Safety Concept; Safety Analysis = deductive requirement analysis and inductive verification
- system view -> Activity = System Design; Safety Analysis = deductive requirement analysis and inductive verification by e.g. FMEA
- EE component view -> Activity = EE Design; Safety Analysis = deductive requirement analysis and inductive quantified analysis
- SW component view -> Activity = SW Design; Safety Analysis = SW architectural analysis

Example for SW development: The the activity devolp the SW-Design from SW-Architekture and SW-Requirments demands an verification of the input.
Parallel to this activitiy a software architectural analysis required. Methods for those analysis are error analysis but also functional analysis. Data flow and control flow analysis are parts of the safety analysis. A potential measure in case of potential risk are data or control flow monitoring. The output of the anlyses integrated into requirements, architecture and design shall be verifified.

The Confirmation Measures take credit from the basic process element as follow:

Functional Safety Audit -> Correct tailoring of that process model in line with ISO 26262 safety lifecycle and correct application during product developmentall

Key questions:

- Do all activities based on the process model?
- Are all inputs verified?
- Are there harmosied safety analysis parallel to the safety activities?
- Are all outputs verified?

- Are all changes processed correctly?

Confirmation Review -> examination of correct verifications and analysis in line with ISO26262 for product development

The following verification patern shall be identifiable

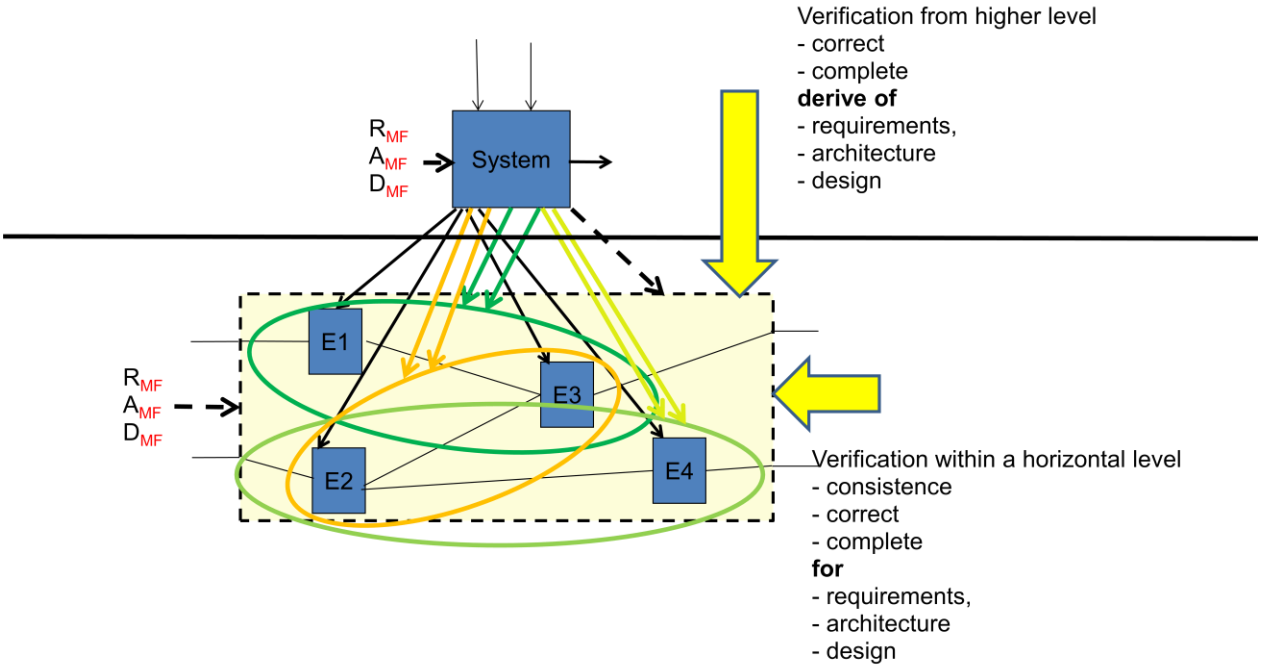

Figure 52: Basic Verification of input and outputs of Safety Activities.

Verifications for inputs from a higher level of abstraction could based on different methodologies. Especially the correctness of derived requirements could based on a methodology comparable with validation. The verification for completeness could based on functional analysis such as FAST (Function Analysis System Techniques).

At the Hardware-Software-Interface (HSI) the verifications could be defined according the following figure:

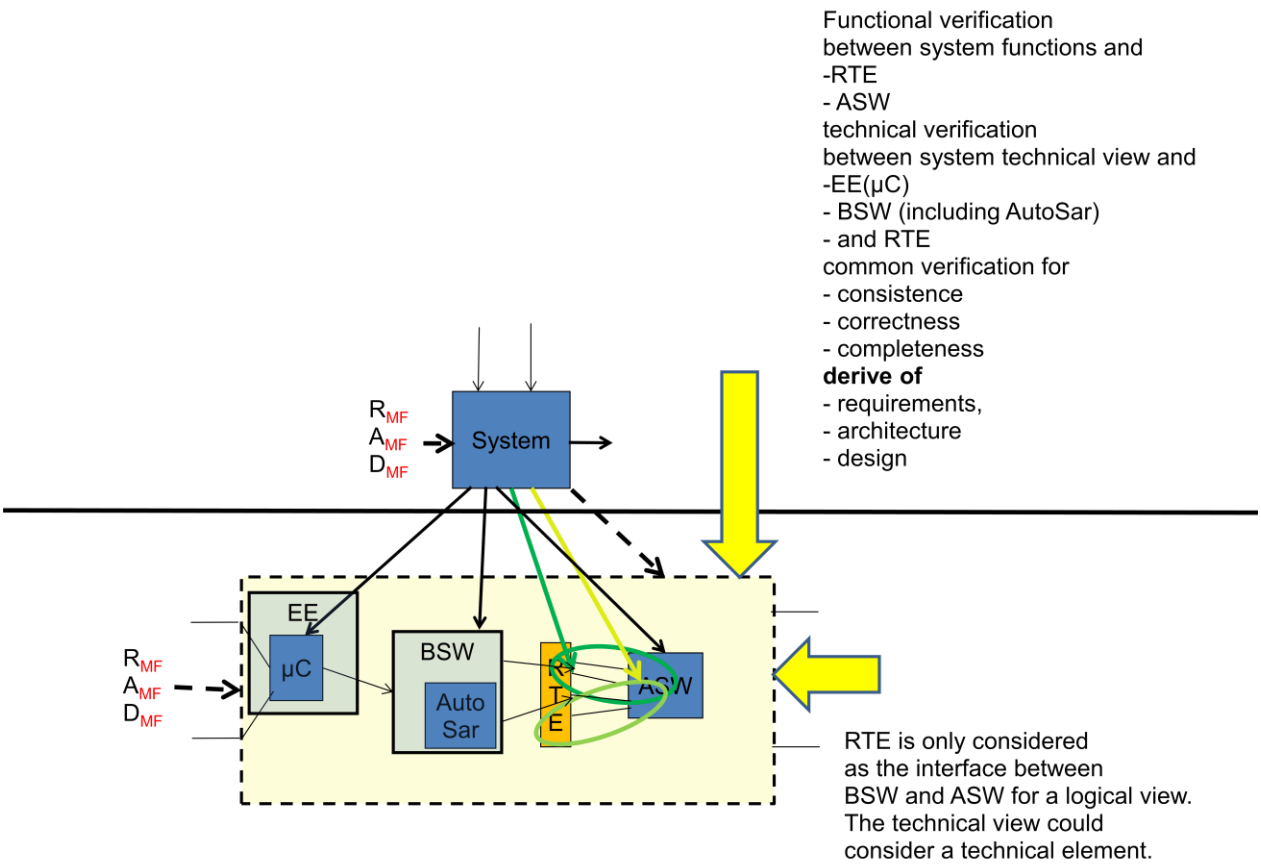

Figure 53: Basic Verification of input and outputs at HSI

Target of the RTE is do design an interface that is independent from the environment for the application SW, so that functional requirements derived from system could be allocated to the application SW.

The conformity of technical requirements from ISO 26262 such as architectural metrics are scope of the safety analysis.

Functional Safety Assessment -> Adequacy and sufficient performance of all safety activities related to given requirements and validation of targets

In case of following of the previous activities, the Functional Safety Assessment based only on the assessing of given outputs (contents of the work-products, including the product itself) from all safety activities. A statement of particial and/or final archievements or sufficient fulfillment of given requirements and targets confirms the fuctional safety of the product.

9.2.8 Common Metrics for evaluation

For each work product, a metric *performance* will be setup rating how well the expectations given in the work product description have been met.

Level 5: beyond expectations described in the Full Project Proposal and evaluation criteria

Level 4: expectation from Full Project Proposal and good level evaluation criteria met

Level 3: expectations not fully met or some evaluation criteria not reached sufficient level but significant improvement achieved

Level 2: no significant improvement achieved or some evaluation criteria are rated incomplete

Level 1: negative impact (performance degraded) and all evaluation criteria are incomplete

This evaluation will be crossed with a metric *industrial interest* qualifying the relevance of the method (or tool or methodology, respectively) covered by the corresponding evaluation scenario.

Level 4: Interesting for evaluation scenario and ready for application in the field

Level 3: Interesting for evaluation scenario but needs to be slightly matured for application in the field

Level 2: Interesting for evaluation scenario but needs to be significantly matured for application in the field

Level 1: Not of interest for the specific evaluation scenario but interesting anyway for application in the field (not considered further for project evaluation – no detailed evaluation result available)

Level 0: Out of scope of evaluation scenario, not of interest for application in the field.

Thus, a graphical representation can be provided for each evaluated work product which gives an interpretation of the industrial potential of the latter.

		Performance				
		1	2	3	4	5
Interest	4	4	8	12	16	20
	3	3	6	9	12	15
	2	2	4	6	8	10
	1	1	2	3	4	5
	0	0	0	0	0	0

Figure 54: Matrix for metrics parameters

10 References

[1] SAFE Report of WT3.6 https://safe.offis.de/svn/svndav/33_WP3_Model_Based_Development/WT3_6_Safety_Code_Generation/Deliverables/SAFE_WT3.6_Report.doc

[2] AUTOSAR; www.autosar.org

[3] Börcsök, Josef, Funktionale Sicherheit, Hüthig Verlag, Heidelberg, 2006.

[4] ISO 26262 (2012) International Organization for Standardization Road Vehicles Functional Safety.

[5] Place, Patrick R.H., Kang, Kyo C, Safety-Critical Software: Status Report and Annotated Bibliography, Technical Report, CMU/SEI-92-TR-5, Software Engineering Institute, Pittsburgh, 1993.

[6] SAFE; www.safe-project.eu

[7] Technische Universität München, SPES 2012 Deliverable D1.1.A, Initiale Modellierungstheorie, München, 2009. Download: http://spes2020.informatik.tu-muenchen.de/results/D-1-1-A.pdf

11 Acknowledgments

This document is based on the SAFE project in the framework of the ITEA2, EUREKA cluster program Σ! 3674. The work has been funded by the German Ministry for Education and Research (BMBF) under the funding ID 01IS11019, and by the French Ministry of the Economy and Finance (DGCIS). The responsibility for the content rests with the authors.

12 figures in this document

13 appendix

13.1 Data Model Documentation of SEP

The Data Model Documentation will be generated by EA. The current document does not contain the export out of EA because the SEP is still under construction. Document D6.b will contain the documentation.