4public
AMALTHEA

(ITEA 2 – 13017)

Enabling of Results from AMALTHEA and others
for Transfer into Application and
building Community around

Deliverable: D 4.1
Gap analysis against ISO 26262

Work Package: 4
Safety

Task: 4.1
Gap analysis of the AMALTHEA development process and tool chain against ISO 26262

Document Type:	Deliverable	**Classification:**	Public
Document Version:	final	**Contract Start Date:**	01.09.2014
Document Preparation Date:	July 9, 2015	**Duration:**	31.08.2017

ITEA 2
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

EUREKA

History

12.03.2015

converted to LaTeX

27.04.2015

added sections to basics-chapter

20.05.2015

added content to gap analysis-chapter

01.07.2015

Major update and review of the entire document.

02.07.2015

Update according to review comments (from TWT)

07.07.2015

Update according to review comments (from OFFIS), fixed several issues, pre-final version

08.07.2015

Minor changes according to review comments (from TWT)

09.07.2015

Final version

Contents

List of Figures

List of Tables

Summary

This document describes gaps of the AMALTHEA platform with regards to the international safety standard ISO 26262 [3] addressing the development of electric/electronic systems for road vehicles. Taking into account the current state of AMALTHEA and based on a thorough analysis of the safety standard, the gaps in AMALTHEA with respect to ISO 26262 were documented in the form of requirements for the AMALTHEA meta-model.

The analysis takes into account the application scope of AMALTHEA and excludes parts of ISO 26262 that are not relevant in this context, like, e.g. hardware development. Work products and their associated requirements defined in the safety standard were aligned with the AMALTHEA meta-model – which we consider to be the core element of the AMALTHEA platform – to identify gaps. Based on the gaps identified, the AMALTHEA meta-model can be adapted and extended to better support the ISO 26262 compliant design and development of safety-relevant embedded software.

1. Basics

1.1. Overview of ISO 26262

The international standard ISO 26262 is a safety standard with the general title "Road vehicles - Functional safety". It was published in 2011, based on the functional safety standard IEC 61508, which deals with safety of electrical and electronic (E/E) systems systems in every area of industry. Instead, ISO 26262 concentrates on the development of E/E systems of passenger cars with a maximum gross vehicle mass up to 3500 kg. It is expected that future versions or extensions of the standard will have a greater coverage and also include vehicles like trucks and motorcycles.

ISO 26262 describes the safety life-cycle of automotive E/E systems, including management, development, production, operation, service and decommissioning. A central element is the hazard analysis and risk assessment in the beginning of the safety life-cycle, where so-called *Automotive Safety Integrity Levels (ASILs)* are defined. Based on this classification, requirements for avoidance of unreasonable residual risk are defined and validation methods are recommended or prescribed.

In addition, intended relations with suppliers and other stakeholders are described, which helps to support integration of an item at product and vehicle level. To this end, the concept of *Safety Elements out of Context* (SEooC) is defined in the standard. Figure 1.1 provides an overview of the different parts of ISO 26262.

1.1.1. Introduction to phases of ISO 26262

The international standard ISO 26262 consists of the following ten parts:

1. Vocabulary
2. Management of functional safety
3. Concept Phase
4. Product development at the system level
5. Product development at the hardware level
6. Product development at the software level
7. Production and operation
8. Supporting processes

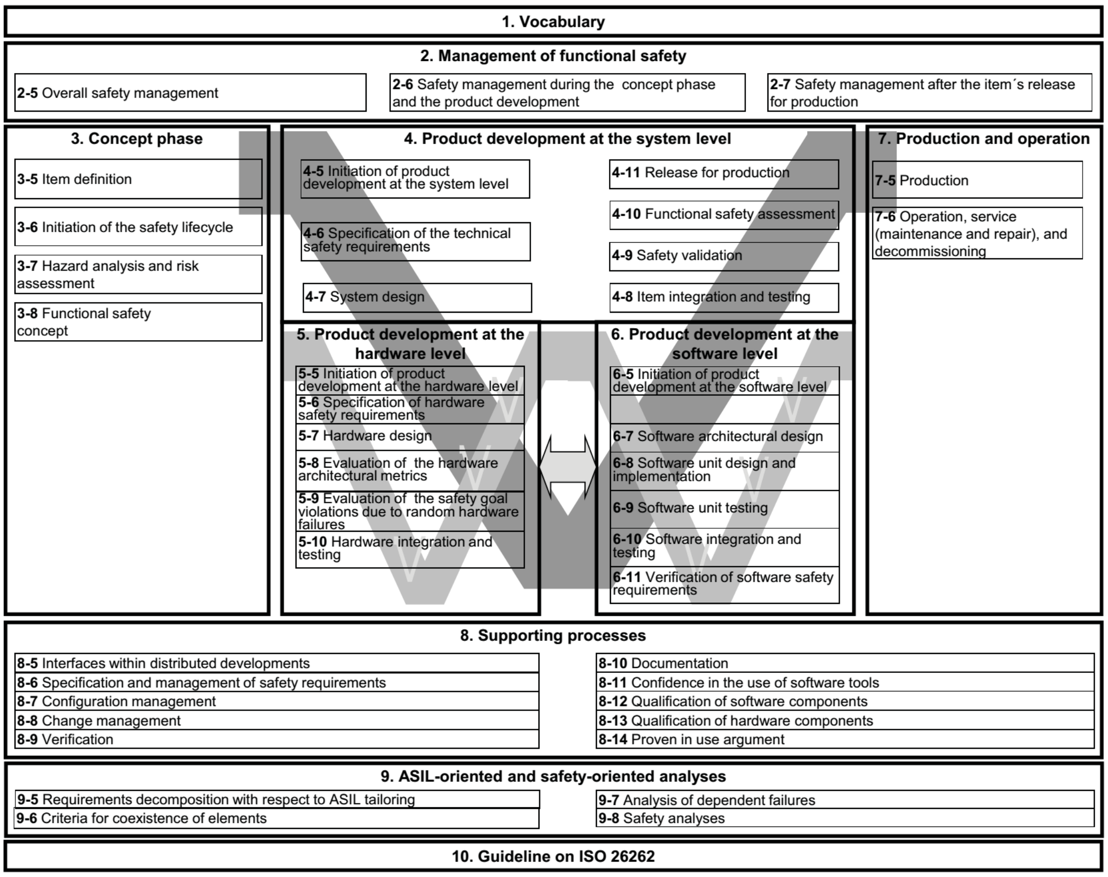
1. Vocabulary
2. Management of functional safety
2-5 Overall safety management
2-6 Safety management during the concept phase and the product development
2-7 Safety management after the item´s release for production
3. Concept phase
3-5 Item definition
3-6 Initiation of the safety lifecycle
3-7 Hazard analysis and risk assessment
3-8 Functional safety concept
4. Product development at the system level
4-5 Initiation of product development at the system level
4-6 Specification of the technical safety requirements
4-7 System design
4-8 Item integration and testing
4-9 Safety validation
4-10 Functional safety assessment
4-11 Release for production
5. Product development at the hardware level
5-5 Initiation of product development at the hardware level
5-6 Specification of hardware safety requirements
5-7 Hardware design
5-8 Evaluation of the hardware architectural metrics
5-9 Evaluation of the safety goal violations due to random hardware failures
5-10 Hardware integration and testing
6. Product development at the software level
6-5 Initiation of product development at the software level
6-6 Specification of software safety requirements
6-7 Software architectural design
6-8 Software unit design and implementation
6-9 Software unit testing
6-10 Software integration and testing
6-11 Verification of software safety requirements
7. Production and operation
7-5 Production
7-6 Operation, service (maintenance and repair), and decommissioning
8. Supporting processes
8-5 Interfaces within distributed developments
8-6 Specification and management of safety requirements
8-7 Configuration management
8-8 Change management
8-9 Verification
8-10 Documentation
8-11 Confidence in the use of software tools
8-12 Qualification of software components
8-13 Qualification of hardware components
8-14 Proven in use argument
9. ASIL-oriented and safety-oriented analyses
9-5 Requirements decomposition with respect to ASIL tailoring
9-6 Criteria for coexistence of elements
9-7 Analysis of dependent failures
9-8 Safety analyses
10. Guideline on ISO 26262

Figure 1.1.: Overview of the ISO 26262 development process (source: [4])

9. ASIL-oriented and safety-oriented analyses

10. Guideline on ISO 26262

Each part is subdivided into different phases. The first part contains a glossary for the used vocabulary and Part 10 is just informative. Thus, in the following we will focus on Parts 2 to 9.

Part 2, management of functional safety, defines which role safety has to play in the overall development process of a product. It is divided into the phases of the overall safety management, safety management during the concept and product development phases as well as safety management after the item's release for production.

Based on the concepts defined in the second part, Parts 3 to 7 adopt the V-model as guideline through the conception and development process of a product, to the point of production and operation. Parts 8 and 9 contain more criteria and rules that have to be applied during the safety life-cycle of a product, for example how requirements have to be defined and which attributes they must have.

1.1.2. V-model of ISO 26262

Parts 3 to 7 of ISO 26262 adhere to the V structure. During concept phase, product development and operation a continuous verification and validation of the design process and the implementation is required.

Part 3 addresses the concept phase and defines the technical safety concept, the item definition, the initiation of the safety life-cycle and hazard analysis and risk assessment. The safety life-cycle covers the entirety of phases from the initial concept to the decommissioning of an item. The role of the hazard analysis and risk assessment is described in the following Section 1.1.3.

The product development at the system level (Part 4) is split into two sections, where first the initiation of the product development at the system level, the specification of the technical safety requirements and the system design have to be performed. This is followed by the parallel product development at hardware and software levels in Parts 5 and 6. Each development process goes from initiation via design and/or implementation to testing, integration and verification of the underlying safety requirements. After hardware and software development, item integration and testing, safety validation, functional safety assessment and release for production are again conducted at system level (Part 4).

Finally, ISO 26262, Part 7 describes the process of production, operation, service and decommissioning.

1.1.3. Hazard Analysis and Risk Assessment and ASIL Assignment

Within the concept phase, a hazard analysis and risk assessment (HARA) needs to be performed. To point out the importance of this method for the rest of the ISO 26262, we briefly describe the recommended process in the following.

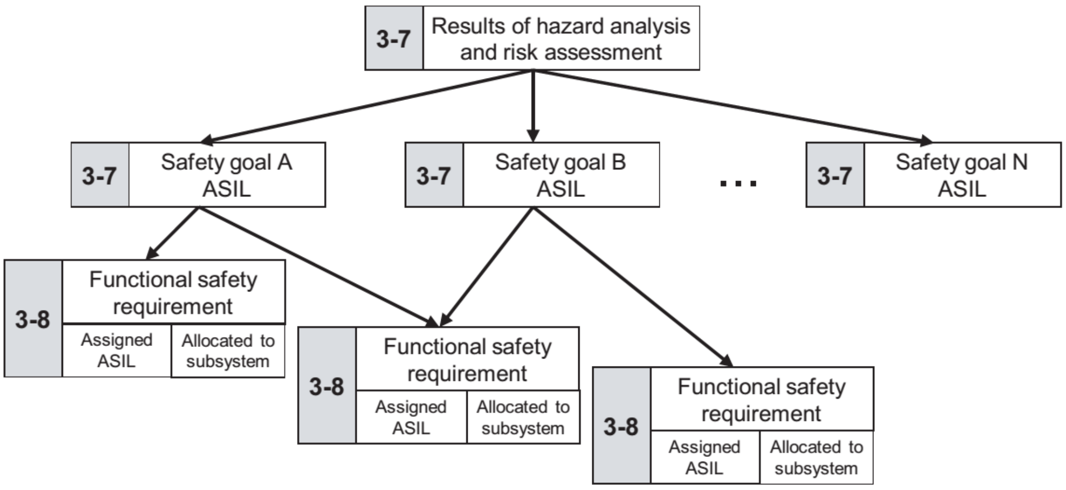

Figure 1.2.: Derivation of Safety Requirements (source: [5])

The development process of an item[1] starts with the concept phase, where the item needs to be defined and relations to other items have to be described. Also the intended behaviour of the item should be defined. Based on this, possibly hazardous events involving the item must be identified and classified. This is already part of the HARA. Depending on the evolved classification, ASILs can be determined. The risk classification using ASILs can be derived by the parameters severity, the exposure and the controllability of a hazardous event. There are four ASILs from the lowest level A to the highest level D, supplemented by the level QM, which stands for (standard) quality management.

To each hazardous event with an ASIL, a *safety goal* should be formulated within the phase of HARA. It has to be verified, that the set of hazardous events is complete, that it complies with the item definition and the determination of ASILs is consistent. From the safety goals, *functional safety requirements* are derived. They inherit the ASILs assigned to the corresponding hazardous event/safety goal. This process is depicted in Figure 1.2 and part of the *functional safety concept*. The functional safety requirements also define safety measures and mechanisms to enable fault detection, failure mitigation and fault tolerance mechanisms and ensure the transition to a safe state (in case of a fault).

The functional safety requirements are later allocated to individual design elements at system, hardware and software levels and refined into so-called *technical safety requirements* taking into account a preliminary system design. This is happening within the first phases of Parts 4, 5 and 6 of ISO 26262, which constitute the kernel of the development of a new product. Every following step in the development of the system at system, hardware or software level shall comply with the allocated safety requirements. An overview of the process and the corresponding parts of the standard is given in Figure 1.3.

[1]An item is representing an E/E system.

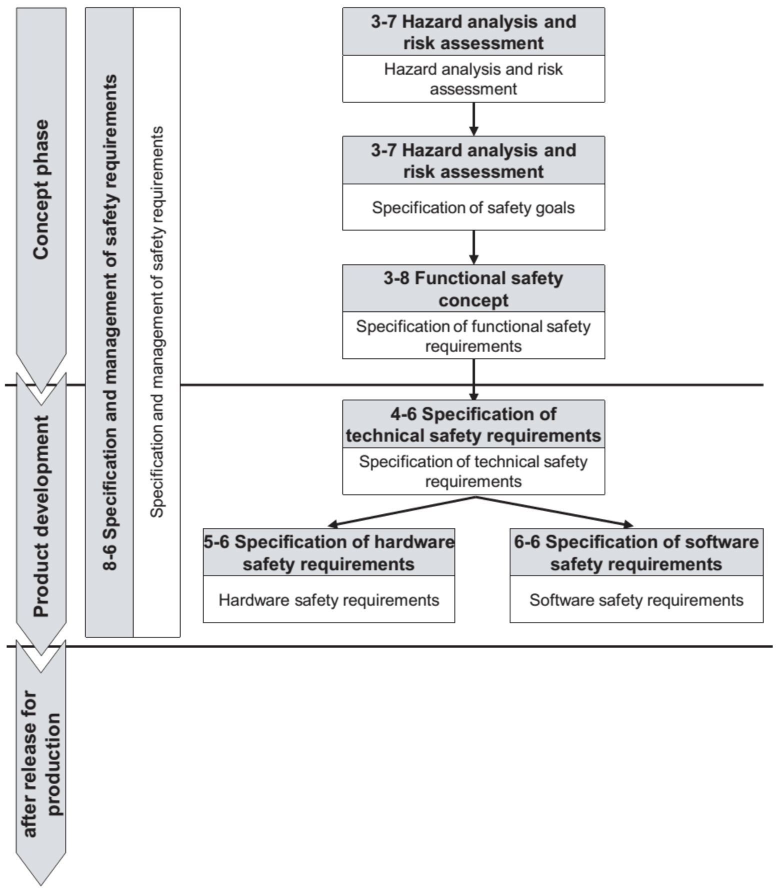
Concept phase
Product development
after release for production
8-6 Specification and management of safety requirements
Specification and management of safety requirements
3-7 Hazard analysis and risk assessment
Hazard analysis and risk assessment
3-7 Hazard analysis and risk assessment
Specification of safety goals
3-8 Functional safety concept
Specification of functional safety requirements
4-6 Specification of technical safety requirements
Specification of technical safety requirements
5-6 Specification of hardware safety requirements
Hardware safety requirements
6-6 Specification of software safety requirements
Software safety requirements

Figure 1.3.: Structure of safety requirements (source: [5])

1.1.4. Safety requirements

As described in Part 8 of ISO 26262, safety requirements have to satisfy certain properties. They have to be allocated to some item or element, having the following characteristics:

- unambiguous and comprehensible
- atomic
- internally consistent
- feasible
- verifiable

and attributes:

- a unique identification
- a status (e.g. proposed, assumed, accepted, reviewed or realised)
- an ASIL

Further, the set of safety requirements shall have a hierarchical structure, which allows traceability, and has to be complete, externally consistent and maintainable. Bi-directional traceability shall be established from safety requirements on different levels of abstraction, to their realisation in the design and the specification of their verification.

There are different types of safety requirements described in ISO 26262:

- functional safety requirements (derived from the safety goals – the top-level requirements – and allocated to the preliminary architectural elements)
- technical safety requirements:
  - hardware safety requirements
  - software safety requirements

Technical Safety Requirements

From the functional safety concept, technical safety requirements are derived. This happens in accordance with further architectural assumptions, e.g. constraints, external interfaces or system configuration requirements. The main task of technical safety requirements is to specify safety-related dependencies between systems or item elements and between the item and other systems. They have to be allocated directly or by further refinement to hardware or software. This means, that the technical safety concept provides a statement of how the functional safety concept is implemented in software or hardware. On the other hand, the system has to comply with the technical safety concept. As shown in Figure 1.3 hardware and software requirements are derived from technical safety requirements depending on their allocation.

The technical safety concept comprises the following aspects that have an impact on both, software and hardware requirements and design:

- specified system and hardware configurations
- hardware-software interface specification
- relevant requirements of the hardware design specification
- external interfaces

Hardware Safety Requirements

Technical safety requirements, which are allocated to hardware, are the basis for hardware safety requirements. Attributes of safety mechanisms, that handle internal and external failures of components and requirements of other components have to be taken into account. Corresponding hardware components inherit the highest ASIL from the hardware safety requirements, that are allocated to it. It has to be ensured, that the traceability between hardware safety requirements, derived requirements and their implementation is maintained down to the lowest level of hardware components. This implies, that we can also have requirements and ASILs on e.g. single processors, cores or memory cells.

Software Safety Requirements

Analogously, software safety requirements are based on the technical safety requirements, which are allocated to software. In addition to the general properties of requirements and the system properties that have an impact on software (as specified in the technical safety concept; see above), further aspects shall be considered:

- timing constraints
- operating modes of the vehicle, system or hardware, that have an impact on the software

Like hardware components, software components have associated ASILs, that are inherited from the highest ASIL of a requirement, that is allocated to the component. Traceability needs to be ensured down to the level of software units.

ASIL Decomposition and Inheritance

It is possible to conduct an ASIL decomposition during the refinement and allocation process of safety requirements. This enables decomposing a safety requirement into redundant architectural elements (represented by derived safety requirements) and to assign a potentially lower ASIL to the decomposed safety requirements. The precise decomposition scheme can be found in Part 9 of the ISO 26262 standard.

Each hardware and software component inherits the highest ASIL of the safety requirements it implements. This implies that the following artefacts are all attributed with an ASIL:

- safety requirements (see above)

- systems and subsystems
- hardware components
- software components
- interfaces

1.1.5. Traceability

According to ISO 26262, Part 8, each safety requirement must be traceable with references to

- its source at the next higher level of abstraction (the requirement or safety goal it was derived from)
- derived safety requirements at a lower level of abstraction or its decomposition (if applicable)
- the realisation of the requirement in the design of the system (if it is a leaf in the "requirements tree")
- the specification of verification (especially, if it is a leaf in the "requirements tree")
- the implementation of safety requirements down to the lowest hardware level (for hardware safety requirements)
- the realisation of safety requirements in software units in the software architectural design (for software safety requirements)

Safety requirements shall be placed under configuration management to enable the tracking of changes and reference baselines. Safety requirements at lower levels of abstraction shall reference a valid baseline of higher level requirements to ensure consistency.

1.1.6. Verification Process

ISO 26262 defines three steps of verification that have to be carried out in each phase of the safety life-cycle:

Verification planning addresses the content of the work products to be verified, the methods used for verification, pass and fail criteria, the environment in which the verification is conducted and the tools used. In addition, actions to be taken if verification fails, need to be defined. The planning should consider adequacy of methods and technologies used for verification, the complexity of the work products under verification and prior experiences. The result of this step is a *verification plan*.

The purpose of the following *verification specification* is to select and specify methods used for the verification and shall include all necessary configurations and test cases in detail (test inputs, execution sequences, environmental conditions, etc.).

Finally, the verification is carried out (*verification execution and evaluation*) according to the verification plan and specification and the results are documented in the *verification report*. The verification specification and report shall clearly reference all relevant requirements (which in turn have to enable tracing of the verification artefacts).

1.1.7. Safety Validation

Validation in the sense of ISO 26262 is conducted to provide evidence that the functional safety concept is appropriate for the functional safety of the developed item. In addition, the correctness and completeness of the safety goals themselves and their complete coverage at vehicle level has to be proven.

The validation step is based on the results on hazard and risk analysis and the verification results at system level after item integration providing the evidence that the safety goals have been implemented correctly. In particular, controllability validation using operating scenarios, validation of the effectiveness of the safety and external measures as well as elements implemented in other technologies (e.g. mechanical) shall be carried out.

1.2. Overview of AMALTHEA

The AMALTHEA tool platform is a model-based open source development environment for automotive systems which was developed in an ITEA2 funded project running from 2011 to 2014. The AMALTHEA platform addresses the following aspects:

- multi-core systems
- product line engineering
- configuration of automotive software systems
- handling of big data volumes
- continuous tool chain
- conformity to standards

The main focus of this document is to examine the last point of this listing with regards to the automotive safety standard ISO 26262.

1.2.1. The AMALTHEA Meta Model

AMALTHEA's main contribution is the meta-model based on Ecore [1] that consists of the following sub-models:

- Hardware model: Describes hardware systems which usually consist of ECUs, microcontrollers, cores, memories, additional peripherals etc.

- Software model: Describes the structure and dependencies of embedded software components including data types, labels representing memory locations, runnables, tasks and activations.
- Constraints Model: Defines different kind of constraints. There are the runnable-sequencing-constraints that can be used to define a required order for the runnables of the software model, the affinity constraints for defining the constraints for the mapping of runnables, processes and schedulers, and the timing constraints for restricting the time span between events or the duration of event chains.
- Event Model: The event model provides the classes to describe system events according to the Best Trace Format (BTF). The model can be used for the tracing configuration, for the modelling of event chains and for some timing constraints.
- Mapping Model: The mapping model is intended to provide tools that use hardware and software models (e.g. code generators) information about the corresponding mappings and allocations. This information contains associations between schedulers and executable software, schedulers and cores as well as data and memories.
- OS Model: Describes the provided functionality of an operating system. It mainly provides a way to specify how access is given to certain system resources. Therefore the concepts of scheduling, buffering, and semaphores are supported.
- Property Constraints Model: The purpose of this model is to limit the design space by providing information about the specific hardware properties that parts of the software rely on, i.e. what properties or features have to be supplied by the respective hardware in order to be a valid mapping or allocation target. This information comprises allocation constraints (describing constraints on cores) and mapping constraints (describing constraints on memories).
- Components Model: Defines systems, their components and interfaces.
- Trace Model: Generic database structure for event traces based on SQLite
- Stimuli Model: Contains stimulus and clock objects. A stimulus is responsible to activate processes and there are different types of it. The stimulus of type Single activates the process only once. The inter-process stimulus defines an activation based on an explicit inter-process activation. The periodic stimulus has an offset time and a recurrence time. The first activation occurs at the offset time. Every following activation occurs after recurrence time. A clock is a time base which describes the progress of time for one or more periodic stimuli in relation to global time. If two equal stimuli have a different time base, the time of task activation can be different.
- Common model: Provides standard elements used by other models such as basic data types.

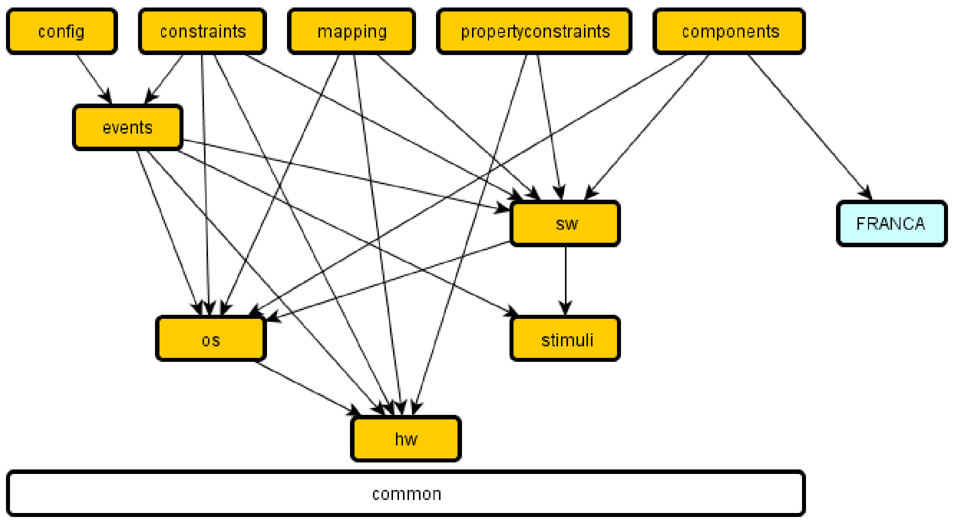

Figure 1.4.: Dependencies of the AMALTHEA models

- Configuration model: Provides common mechanisms for configuration purposes. Currently this includes solely event configurations.
- Requirements model: AMALTHEA already provides the opportunity to define requirements based on external tools such as ProR [2]. However, currently there is no direct integration of requirements in the AMALTHEA meta model. While referencing AMALTHEA model elements from external tools is certainly feasible, a feature for bi-directional links is still missing and thus we assume a requirements model will be added in the course of the AMALTHEA4public project. In the following, we will thus mention this model although it is not yet implemented in the platform. Some of gaps described in the following Chapter 2 can be closed by an appropriate requirements model.

The relationship between the individual models is shown in the diagram in Figure 1.4. It is evident that the common model provides standard elements used by other models as containments. Franca is a common interface definition language that can be used in component models.

1.2.2. Workflow using the AMALTHEA platform

AMALTHEA does not advocate a single workflow to be used in embedded systems development. Instead, the AMALTHEA meta model opens up a plethora of possibilities for individual workflows and can be used as common ground for interfacing of development tools. AMALTHEA establishes an open source ecosystem that enables software and system developers as well as tool vendors to realise their own workflows. In the following a few examples of applying AMALTHEA are given:

- partitioning of software components/designs for embedded multi-core platforms

- mapping of software to hardware taking into account constraints
- simulation of embedded systems timing behaviour (e.g. on multi-core architectures)
- (technical) requirements engineering, variant and architecture modelling
- component behaviour modelling
- generating code templates

In all of the above applications, AMALTHEA models are used for maintaining the relevant information and exchanging information between tools. In addition, AMALTHEA models can be used to directly specify system architecture and behaviour at an abstract level.

2. Gap Analysis

The ISO 26262 safety standard specifies processes and work products for the development of automotive E/E systems. As we pointed out in Section 1.2, the core of the AMALTHEA platform is the corresponding meta model that forms a common basis for tooling in software design, development and analysis for embedded (multi-core) systems. The following gap analysis will thus focus on this core concept and investigate where AMALTHEA work products/meta models are already conformant with the work products and requirements defined by the ISO 26262 standard and where there are gaps or inconsistencies in the current AMALTHEA meta model.

Please note, that the goal of this gap analysis is *not* to check whether AMALTHEA provides the tooling of a full-fledged ISO 26262 workflow. This will never be the case and is not the goal of the AMALTHEA platform. Instead, we want to ensure that AMALTHEA *enables* ISO 26262 conformant developments. This requires the addition of (external) tooling which (partly) uses the AMALTHEA platform. However, if ISO 26262-relevant information is missing in AMALTHEA models (like ASILs on software components), it will be more difficult to establish an integrated ISO 26262 development process using these models as missing information needs to be maintained externally.

2.1. Overview of Important Work Products

As a first step in this gap analysis we categorise the work products of ISO 26262 according to their relevance in the AMALTHEA context. As mentioned before, our focus lies on the meta-model of AMALTHEA, e.g. we want to analyse, which safety-related requirements are already met by the model. It is out of the scope of this document to describe intended proceedings to meet all requirements given in the standard.

Work products categorised as *included* are already or should be covered or represented by the AMALTHEA platform to better support its use (or the use of the addressed meta-models) in an ISO 26262 conformant development workflow. *Partially included* work products have an impact on AMALTHEA but need not to be represented in their entirety. Finally, some work products are of no relevance for the current scope of the AMALTHEA platform and are therefore *excluded* in this gap analysis. These might be considered later, if necessary.

Some work products require other work products as prerequisites. For example, we first need to establish the *functional safety concept* as well as the *validation plan* before we are able to generate the *technical safety requirements specification.* This would need to be considered in a *requirements meta-model.* So far, no requirements meta-model is defined in AMALTHEA. But in the scope of the Amalthea4Public project it is envisaged to be established. Thus, in the following we also take into account ISO 26262 requirements on

Part	Work Product	Relation to AMALTHEA
3.5	Item definition	Necessary for many other work products; the item definition is referenced by safety requirements which are in turn referring to their realisation (and the other way round). Impact on requirements and software models as requirements and software artefacts are related to an item.
3.7	Safety goals	Need to be represented in a requirements model (top-level requirements)
3.8	Functional safety concept	Relevant for requirements model
4.6	Technical safety requirements specification	Relevant for requirements and software models
4.7	System design specification	Components model
4.7	Technical safety concept	Relevant for requirements and software models
5.6	Hardware safety requirements specification	Relevant for a complete requirements model
5.6	Hardware-software interface specification	Relevant for hardware, software and requirements models
5.7	Hardware design specification	Represented in the AMALTHEA hardware model
6.6	Software safety requirements specification	Represented in a requirements model and allocated to software components and thus relevant for the software model as well.
6.7	Software architectural design specification	Represented in the software model
6.8	Software unit design specification	Relevant for software model
9.5	Update of architectural information	Requirements, software and hardware models; it must be traceable where ASIL decomposition has taken place.
9.5	Update of ASIL as attribute of safety requirements and elements	Requirements, software and hardware models; it must be traceable where ASIL decomposition has taken place.
9.6	Update of ASIL as attribute of sub-elements of elements	Requirements, software and hardware models; indication/justification where sub-elements that are safety-related and sub-elements without ASIL coexist for the same element.

Table 2.1.: ISO 26262 work products *included* in the gap analysis

Part	Work Product	Relation to AMALTHEA
3.7	Hazard analysis and risk assessment	Basis for the safety goals (top-level requirements) and thus should be referenced in the requirements model.
3.7	Verification review report of the hazard analysis and risk assessment and the safety goals	The requirements model should reference the relevant report.
3.8	Verification report of the functional safety concept	References from the requirements model
4.5, 4.6, 4.9	Validation plan	References for validation activities of system elements (component model)
4.7	Safety analysis report	Might be referenced in the component model
4.7	System verification report	Can be referenced from component model
5.7	Hardware safety requirements verification report	Evidence that the hardware safety requirements are consistent and complete, relevant for requirements and hardware models. Reference to the information should be sufficient.
6.7	Dependent failures analysis report	Analysis of dependent failures on software level; impact on software model (see also 9.7)
6.8	Software unit implementation	Impact on software model but not part of the software model itself
6.10, 6.11	Software verification plan	References to verification plan in software model
6.10, 6.11	Software verification report	References to verification report in software model
6.10, 6.11	Software verification specification	References to verification spec. and test cases in software model
8.9	Verification plan	References from the requirements model, see also ISO 26262-3.8.4.5
8.9	Verification specification	References from the requirements model, see also ISO 26262-3.8.4.5
8.9	Verification report	References from the requirements model, see also ISO 26262-3.8.4.5
8.12	Software component documentation	Documentation can be referenced in the software model
8.12	Software component qualification report	Support attributes to indicate software components' qualification
9.7	Analysis of dependent failures	Impact on requirements, component and software models; redundant elements in the architecture need to be identified and checked for their independence.

Table 2.2.: ISO 26262 work products *partially included* in the gap analysis

a requirements meta-model. This implies that address, e.g. the *functional safety concept* and the *technical safety requirements specification*, because their definition may influence attributes of requirements defined within such a requirements meta-model.

Table 2.1 lists the work products of ISO 26262 that have to be represented within the AMALTHEA meta-model and are thus included in the gap analysis. Table 2.2 lists work products that have an impact on the meta-model but are not represented in their entirety. The work products not considered in the gap analysis, because they are currently out of scope for AMALTHEA, are documented in the Appendix in Section A.1.

Additionally, clauses of Part 8 of ISO 26262 concerning the management of safety requirements need to be considered. These clauses do not define work products themselves.

2.2. Gap Analysis for Included Work Products

This section comprises the analysis, which parts of the ISO 26262 are not yet addressed within the AMALTHEA platform but are deemed important for safety-relevant applications of AMALTHEA. To this end, gaps are formulated in terms of requirements with respect to the AMALTHEA meta-model. Based on the identified gaps we propose to adapt/extend the AMALTHEA meta model such that

1. users of the AMALTHEA tool platform have improved support of ISO 26262 directly integrated into the meta-model, and
2. reviewers and safety managers are given better support in verifying requirements of ISO 26262 for a given model.

The analysis is structured by grouping the work products logically. For every such group we will give a description of the required content and provide a list of all requirements from the ISO 26262 standard, that have to be met to define this work product in AMALTHEA. For traceability we will also give a list of all work products required as inputs.

As motivated before, we will later define requirements concerning AMALTHEA. If necessary, they are followed by some notes providing a rationale or other details.

2.2.1. Item definition

The work product *item definition* of ISO 26262 is the basis for the safety-life-cycle of a product. It describes the properties of the developed item, how the item is integrated into the vehicle and which dependencies between the item and its environment exist.

Table 2.3: Work Products and Requirements for the Item Definition

Work product	Result of	Required Work Products (Input)
Item Definition	3.5.4.1 3.5.4.2	None

It shall be possible to conduct a hazard analysis and define safety goals based on the item definition. This means, that requirements management should be extended in some way. Thus, we define the following requirements with respect to the AMALTHEA platform[1]:

Table 2.4: AMALTHEA Requirements: Item Definition

ID	Ref.	Description
ID.1	3.5.4.1 3.5.4.2	The item definition has to be defined in an ISO 26262-compliant way, including the information given in 3.5.4.1 and 3.5.4.2. These are functional and non-functional requirements of the developed item and dependencies between the item and its environment. To be more concrete, the following information has to be provided: • functional concept • operational and environmental constraints • legal requirements • behaviour achieved by similar functions, items or elements • behaviour expected from the item • potential consequences of behaviour shortfalls including known failure modes and hazards In addition, the following attributes of the item and its interfaces shall be defined: • the elements of the item • assumptions concerning the effects of the item's behaviour on other items or elements • functionality required by/from other items, elements and the environment • the allocation and distribution of functions among the involved systems and elements • the operating scenarios which impact the functionality of the item

[1]The reference column refers to the relevant requirements of ISO 26262. E.g. 3.5.4.1 refers to requirement 5.4.1 of ISO 26262 Part 3. If an entry is referenced that has sub-requirements, all these are assumed to be included.

2.2.2. Hazard Analysis and Risk Assessment

The importance of the hazard analysis and risk assessment (HARA) is already described in Section 1.1.3. We do not consider HARA to be an integral part of AMALTHEA models (in fact it has been categorised to be partially included; see Table 2.2) it should be traceable and thus has an impact, in particular on the requirements meta-model. Thus, we do not state any requirements for AMALTHEA in this section, but instead HARA will be impacting the requirements for other work products (to be) integrated into the AMALTHEA meta-model. For example, we will need to define hazards, i.e. malfunctional behaviour of the system. Faults and failures need to be described and categorised by, e.g. their rate of occurrence and their impact to the whole system. They correspond to technical elements as hardware and software and will be described in later work products.

Note, that in the following all work products marked with * will just be included partially in this gap analysis. Elements that are out of scope and are excluded are marked with †.

Table 2.5: Work Products and requirements for the HARA

Work product	Result of	Required Work Products (Input)
Hazard analysis and risk assessment*	3.7.4.1 3.7.4.2 3.7.4.3 3.7.4.4.1 3.7.4.4.2	Item Definition
Validation plan*	4.5.4.2 4.6.4.6.2 4.9.4.2	Safety goals Hazard analysis and risk assessment* Functional safety concept Functional safety assessment plan† Safety plan† Project plan†

2.2.3. Functional Safety Concept

In this section we will analyse AMALTHEA with respect to the functional safety concept defined in ISO 26262. In Section 1.1.4 we already described the hierarchy of safety requirements derived from the safety goals. Functional safety requirements are refined to technical safety requirements allocated to hardware and software. The following table list the work products, the associated ISO 26262 requirements and the work products' prerequisites in the safety life cycle.

Table 2.6: Work Products and Requirements for the Functional Safety Concept

Work product	Result of	Required Work Products (Input)
Safety Goals	3.7.4.4.3 3.7.4.4.4 3.7.4.4.5 3.7.4.4.6	Item definition
Functional Safety Concept	3.8.4.1 3.8.4.2 3.8.4.3 3.8.4.4.1	Item definition Hazard analysis and risk assessment* Safety goals
Verification report of the functional safety concept* Verification plan* Verification specification* Verification report*	3.8.4.5.1 8.9.4	Item definition Hazard analysis and risk assessment* Safety goals

As mentioned before, in AMALTHEA it is possible to define requirements based on the external tool ProR [2].

Requirements do not necessarily need to be safety-related, but currently there is no possibility to mark requirements as safety-related. Furthermore, the hierarchical structure provided by AMALTHEA is not sufficient to fully support ISO 26262 conformant requirements management. Traceability of requirements must be established within of the hierarchical structure as well as enabling allocation of requirements to system elements. The latter is discussed in later sections of this document.

In addition, the expressiveness of certain requirement types needs to be enhanced, and refinement of requirements must be supported. In the following we begin with the requirement type *safety goal* as top-level requirement.

Table 2.7: AMALTHEA Requirements: Safety Goals

ID	Ref.	Description
SG.1	3.7.4.4.3	Safety goals need to be designated as top-level safety requirements. Additional requirements that can be indirectly derived: • It shall be possible to categorise requirements using an hierarchical structure: Safety goals, functional safety requirements, technical safety requirements and at the same level of hierarchy hardware and software safety requirements. • It shall be possible to distinguish between safety-related and non-safety-related requirements.
SG.2	3.7.4.4.5	Safety goals shall define the safe state(s) that is/are to achieve.

Functional safety requirements have to be derived from the safety goals. It is very important to allow inheritance of all important information of the safety goals to the requirements. If possible, functional safety requirements should already be formulated using the corresponding terms and concepts of the AMALTHEA meta-model. This eases successive allocation processes.

Table 2.8: AMALTHEA Requirements: Functional Safety Concept

ID	Ref.	Description
FS.1	3.8.4.2.1	Every functional safety requirement needs to be derived from a safety goal.
FS.2	3.8.4.2.2	At least one functional safety requirement shall be specified for each safety goal.
FS.3	3.8.4.2.1 3.8.4.2.2	There has to be the possibility to link functional safety requirements with the safety goals from which they are derived.
FS.4	3.8.4.2.3	It shall be possible to express the following properties for functional safety requirements: • operating modes • fault tolerant time intervals • safe states • emergency operation intervals • functional redundancies
FS.5	3.8.4.2.3 3.8.4.2.4 3.8.4.2.5 3.8.4.2.6	Transitions to and from safe states and further information about safe states shall be specified as functional safety requirements or shall be expressed by them.
FS.6	3.8.4.2.6	It shall be possible to express necessary actions of the driver as functional safety requirements.
FS.7	3.8.4.3.1 3.8.4.3.2 3.8.4.3.3	Functional safety requirements shall be allocated to corresponding architectural structures (as e.g. hardware, software, interfaces). If they are realised by elements of other technologies (e.g. mechanical) or rely on external measures, additional functional safety requirements shall be derived. In addition, functional safety requirements for the interfaces to the elements of other technologies or implemented by external measures shall be specified.
FS.8	3.8.4.3.1	Architectural elements as hardware, software or interfaces shall have an ASIL attribute.

Notes

- FS.1, FS.2: These two requirement are necessary to ensure, that the set of all safety requirements is complete. It is allowed to define more than one functional safety requirement per safety goal and also to link more than one safety goal to a functional safety requirement.

- FS.3: The hierarchy of safety requirements must be traceable.
- FS.4: This implies, that the following is expressible:
  - Failures, failures modes and their properties (e.g. multi-point failures, single-point failures, dependent and independent failures, random hardware failures, systematic failures, failure rates)
  - Faults, fault models and their properties (e.g. multi-point faults, single-point faults, perceived and permanent faults, residual fault, systematic faults or transient faults)
  - The warning and degradation concept; further information is given in the system design.
- FS.6: This is part of the warning and degradation concept.
- FS.7: The behaviour of the ASILs of interacting subsystems or system elements is described in the management of safety requirements.
- FS.8: ASILs are attributed down to the lowest level of software or hardware units. For AMALTHEA this means, e.g. for hardware, that the systems, ECUs, micro-controller and cores shall have ASILs. In the AMALTHEA context these ASILs are given by the hardware vendor.

2.2.4. System Design

The system design covers all required processes including software and hardware development. Product development at system, hardware and software level (ISO 26262 Parts 4, 5 and 6) form the core part of the ISO 26262 V-model. Therefore we need to be able to represent a large variety of related work products in AMALTHEA, even if we exclude V&V processes and item integration.

As described in Section 2.2.5, it is highly recommended to represent hardware safety requirements and to allow allocation of requirements and ASILs to hardware to achieve ISO 26262 compliance. The technical safety concept and the system design defines the interface between software, hardware and the overall system. We have to investigate, which parts of it are already represented sufficiently. Again, we give the full list of required work products.

In this section we also consider the *ASIL decomposition*, the *coexistence* of elements with different ASILs at the same level of abstraction and the analysis of dependent failures. These steps are executed at system, hardware or software level and are optional steps in the development process.

Table 2.9: Work Products and Requirements for the System Design

Work product	Result of	Required Work Products (Input)
Technical safety requirements specification	4.6.4.1 4.6.4.2 4.6.4.3.1 4.6.4.4 4.6.4.5.1	Validation plan* Functional safety concept
Technical safety concept	4.7.4.1 4.7.4.2 4.7.4.3 4.7.4.4 4.7.4.5	Technical safety requirements specification Item integration and testing plan†
System design specification	4.7.4.1 4.7.4.2 4.7.4.3 4.7.4.4 4.7.4.5	Technical safety requirements specification Item integration and testing plan†
Hardware-software interface specification	4.7.4.6 5.6.4.10 5.6.4.11 6.6.4.4	System design specification Technical safety concept Safety plan† Item integration and testing plan† Technical safety requirements specification Software verification report* Software verification plan* Software safety requirements specification
Update of architectural information	9.5.4	At the level of ASIL decomposition: • safety requirements • architectural information
Update of ASIL as attribute of safety requirements and elements	9.5.4	At the level of ASIL decomposition: • safety requirements • architectural information

Table 2.9: Work Products and Requirements for the System Design

Work product	Result of	Required Work Products (Input)
Update of ASIL as attribute of sub-elements of elements	9.6.4	At the level of which the coexistence analysis is performed: • safety requirements • architectural information
System verification report* Verification plan* Verification specification* Verification report*	4.6.4.6 4.7.4.8	Validation plan* Functional safety concept Item integration and testing plan Technical safety requirements specification
Analysis of dependent failures*	9.6.4	At the level of application (system, hardware or software): • independence requirements† • freedom from interference requirements† • architectural information

The hierarchy of safety requirements in the AMALTHEA model is not yet sufficient to achieve compliance to ISO 26262. We formulate the resulting requirements for the technical safety requirements in the following.

We need to represent additional properties of system elements. As safety requirements can be violated by malfunctional behaviour of system elements, we have to be able to model these failures and the faults, that cause them. Modelling in this context means, that we have to represent the different types of faults and failures described in ISO 26262, their occurrence and some further attributes.

Table 2.10: AMALTHEA Requirements: Technical Safety Requirements Specification

ID	Ref.	Description
TSR.1	4.6.4.1.1	Technical safety requirements need to be designated as such.
TSR.2	4.6.4.1.1	Technical safety requirements shall be derived from functional safety requirements.
TSR.3	4.6.4.1.1	It shall be possible to express external interfaces, constraints and system configuration requirements as a technical safety requirement.

Table 2.10: AMALTHEA Requirements: Technical Safety Requirements Specification

ID	Ref.	Description
TSR.4	4.6.4.1.1	There has to be the possibility to link technical safety requirements with the functional safety requirements, from which they are derived.
TSR.5	4.6.4.2.2	Technical safety requirements need to specify safe states and safety mechanisms derived from the functional safety requirements.
TSR.6	4.6.4.2.2	It must be possible to express measures to detect, indicate and control faults of the system, external devices that interact with the system as well as measures to enable the system to achieve and preserve safe states. This includes measures described in the warning and degradation concept, i.e. necessary actions of the driver.
TSR.7	4.6.4.2.3	For safety mechanisms the following information shall be specified: • the transition to the intended safe state • the fault tolerant time interval • the emergency operation interval, if the intended safe state cannot be reached immediately • the measures to maintain the intended safe state
TSR.7	4.6.4.4.3	Failure rates of hardware components as well as probabilities of hazardous events shall be expressible.

Notes

- TSR.7: As already mentioned in Section 2.2.3 this implies that faults and failures are expressible. Furthermore, the corresponding detection intervals are needed.
- TSR.8: To manage faults, failures and their respective consequences for the safety goals, the rate of their occurrence must be included to the model.

The mapping model of AMALTHEA already provides the possibility to allocate software units to hardware components. Memory access, core allocation, tracing of runnables and further aspects can be represented. The meta-model allows for hardware and software to be broken down into sufficiently small components.

ISO 26262 demands the allocation of technical safety requirements to corresponding system elements. This, and the possibility to define any kind of connection between system elements (e.g. containedness or interfaces), requires tracing of requirements at arbitrary levels of granularity. For this, it is important to define allocation mechanisms in such a way, that bidirectional tracing becomes feasible.

Table 2.11: AMALTHEA Requirements: Technical Safety Concept

ID	Ref.	Description
TSC.1	4.7.4.1.2 4.7.4.5.1	Technical safety requirements shall be allocated at corresponding system elements.

Table 2.12: AMALTHEA Requirements: System Design Specification

ID	Ref.	Description
SD.1	4.7.4.2.2	System elements down to lowest level of abstraction shall be attributed with an ASIL. This also includes, e.g. interfaces and software units. The element inherits the highest ASIL from the requirements, that it implements.
SD.2	4.7.4.2.3 9.6.4	If a system element is comprised of sub-elements with different assigned ASILs, each of these shall be treated in accordance of the highest ASIL, unless the sub-elements' independence can be proven.
SD.3	4.7.4.2.4	Internal and external interfaces of safety-related system elements have to be defined.

Note In particular, the partitioning and mapping processes need to be traceable down to the lowest level.

Table 2.13: AMALTHEA Requirements: Hardware-Software Interface Specification

ID	Ref.	Description
HSI.1	4.7.4.6.1	A hardware-software interface specification shall be defined.
HSI.2	4.7.4.6.1	Hardware, that is controlled by software, must be traceable to this software.
HSI.3	4.7.4.6.1	Technical safety requirements shall be linked to hardware-software interfaces.
HSI.4	4.7.4.6.2	It must be possible to define timing constraints, memory accesses, operating modes of hardware and shared and exclusive use of hardware resources.
HSI.5	4.7.4.6.3	Relevant diagnostic capabilities of hardware must be specified.
HSI.6	6.6.4.4	The hardware-software interface specification shall be specified at a level of detail allowing the correct control and usage of the hardware. In particular, safety-related dependencies between hardware and software shall be described.

Notes

- HSI.2: This requirement seems to be already partly fulfilled by the AMALTHEA mapping model.
- HSI.4: This is required during hardware design. Parts of it, however, apply to the AMALTHEA hardware model. Some aspects of this requirement seem to be already fulfilled.

For ASIL decomposition it is not sufficient to represent hierarchical structures of system elements or interfaces. The data model of AMALTHEA must be extended, such that dependent failures can be specified correctly. In particular, it shall be possible to represent *independent* and *redundant* elements as well as *freedom of interference*.

Table 2.14: AMALTHEA Requirements: ASIL Decomposition / Update of Architectural Information

ID	Ref.	Description
ADA.1	9.5.4.3 9.5.4.6 9.5.4.7 9.5.4.11	It shall be possible to explicitely express independence and redundancy of architectural elements.
ADA.2	9.5.4.6 9.5.4.7	Decomposition of architectural elements at hardware or software level shall be traceable back to system level.
ADA.3	9.5.4.3	It shall be possible to represent the ASIL decomposition of architectural elements (at hardware, software or system level) explicitly.

Notes

- ADA.1 This is related to expressing failures and faults in the functional safety concept.

Table 2.15: AMALTHEA Requirements: ASIL Decomposition / Update of ASIL as Attribute of Safety Requirements

ID	Ref.	Description
ADR.1	9.5.4.3 9.5.4.4	It shall be possible to represent ASIL-decomposed safety requirements explicitly. This decomposition shall be distinct from the derivation relation of safety requirements as they are refined.

Table 2.16: AMALTHEA Requirements: Coexistence of Elements / Update of ASIL as Attribute of Sub-Elements of Elements

ID	Ref.	Description
CE.1	9.6.4.4 9.6.4.5	It shall be possible to express non-interference of system elements.
CE.2	9.6.4.5	It shall be possible to express coexistence of system elements.

Notes

- CE.1, CE.2: These requirements are relevant especially for mapping processes, e.g. if software components with different ASILs run on the same microcontroller.

2.2.5. Hardware Design

Product development at hardware and software level shall be executed in parallel inside the inner V of the ISO 26262 V-model. In different phases of development, verification and validation activities shall be conducted. There is an iterative exchange of information between these two central parts of the development process and the system level. The hardware-software interface specification at system level allows to connect different components. In the AMALTHEA meta-model this can already be realised by using the mapping model.

But to guarantee, that safety goals are satisfied down to the lowest level of abstraction of a system, we need to supply safety-relevant attributes (e.g. ASILs) to hardware, even if AMALTHEA is not used to develop it. This information is necessary to perform validation and verification at system level.

Table 2.17: Work Products and Requirements for the Hardware Design

Work product	Result of	Required Work Products (Input)
Hardware safety requirements specification	5.6.4.1 5.6.4.2 5.6.4.3 5.6.4.4 5.6.4.5 5.6.4.6 5.6.4.7 5.6.4.8	Hardware-software interface specification Technical safety concept Safety plan† System design specification
Hardware Design Specification	5.7.4.1 5.7.4.2	Hardware safety requirements specification Hardware-software interface specification Safety plan† System design specification
Hardware safety requirements verification report*	5.6.4.9	Hardware safety requirements specification Technical safety concept Hardware-software interface specification Safety plan† System design specification

As before, we need to address the hierarchical structure of hardware safety requirements.

Table 2.18: AMALTHEA Requirements: Hardware Safety Requirements Specification

ID	Ref.	Description
HWR.1	5.6.4.1	Hardware safety requirements shall be designated as such.
HWR.2	5.6.4.1	Hardware safety requirements shall be derived from the technical safety requirements allocated to hardware.
HWR.3	5.6.4.1	There shall be the possibility to link hardware safety requirements to the technical safety requirements, from which they are deduced.

The allocation of safety requirements at hardware elements was already addressed in Section 2.2.4. The following requirements focus especially the hardware safety requirements.

Table 2.19: AMALTHEA Requirements: Hardware Design Specification

ID	Ref.	Description
HWD.1	5.7.4.1.1 5.7.4.1.2	Hardware safety requirements shall to be allocated to the hardware components implementing them.
HWD.2	5.7.4.1.5 5.7.4.5.3	Traceability needs to be ensured down to the lowest level of hardware components represented in AMALTHEA.

Notes

- HWD.1: For this it may also be necessary to express hardware architectural metrics (single-point fault metrics, latent fault metrics, etc.)

2.2.6. Software Design

Regarding the focus of the existing AMALTHEA tool platform, software design is one of the most relevant topics of ISO 26262 in this analysis. We restrict our analysis to work products of clauses 6, 7, 8 and 10, as software unit testing and V&V processes are not in our scope.

Table 2.20: Work Products and Requirements for the Software Design

Work product	Result of	Required Work Products (Input)
Software safety requirements specification	6.6.4.1 6.6.4.2 6.6.4.3 6.6.4.5	Technical safety concept Safety plan† Hardware-software interface specification System design specification

Table 2.20: Work Products and Requirements for the Software Design

Work product	Result of	Required Work Products (Input)
Software architectural design specification	6.7.4.1 6.7.4.2 6.7.4.3 6.7.4.4 6.7.4.5 6.7.4.6 6.7.4.9 6.7.4.10 6.7.4.14 6.7.4.15 6.7.4.17	Safety plan† Hardware-software interface specification Software safety requirements specification System design specification Design and coding guidelines for modelling and programming languages†
Software unit design specification	6.8.4.2 6.8.4.3 6.8.4.4	Safety plan† Software safety requirements specification Software architectural design specification Design and coding guidelines for modelling and programming languages†
Software unit implementation*	6.8.4.4	Safety plan† Software safety requirements specification Software architectural design specification Design and coding guidelines for modelling and programming languages†
Embedded Software		
Software component documentation*	8.12.4.3.1	
Software component qualification report*	8.12.4.3.5	

Table 2.20: Work Products and Requirements for the Software Design

Work product	Result of	Required Work Products (Input)
Software verification plan* (at unit, integration and software safety requirements level)	6.5.4.1	Software safety requirements specification
	6.5.4.2	Item integration and testing plan†
	6.5.4.3	Software design specification
	6.5.4.4	Technical safety concept
	6.5.4.7	Software architectural design specification
	6.6.4.6	Software unit implementation
	6.9.4.2	Hardware-software interface specification
	6.9.4.3	Software architectural design specification
	6.9.4.4	Software verification specification*
	6.9.4.5	Software verification report*
	6.9.4.6	Safety plan†
	6.10.4.1	Project plan†
	6.10.4.2	Integration testing report†
	6.10.4.3	
	6.10.4.4	
	6.10.4.5	
	6.10.4.6	
	6.10.4.8	
	6.11.4.1	
	6.11.4.2	
	6.11.4.3	
	6.C.4.2	
	6.C.4.4	
	6.C.4.7	
	6.C.4.10	
Software verification specification* (at unit, integration and software safety requirements level)	6.6.4.7	Software safety requirements specification
	6.6.4.8	Software unit design specification
	6.9.4.4	Software architectural design specification
	6.9.4.6	Software verification plan*
	6.10.4.1	Software verification report*
	6.10.4.2	Safety plan†
	6.10.4.4	Integration testing report†
	6.10.4.5	Software unit implementation*
	6.10.4.7	Hardware-software interface specification
	6.10.4.8	
	6.11.4.1	
	6.11.4.2	
	6.11.4.3	

Table 2.20: Work Products and Requirements for the Software Design

Work product	Result of	Required Work Products (Input)
Software verification report*	6.6.4.7 6.6.4.8 6.9.4.2 6.10.4.2 6.11.4	Software safety requirements specification Software unit design specification Software architectural design specification Software verification plan* Software verification specification* Software verification report* Safety plan† Integration testing report† Software unit implementation* Hardware-software interface specification
Update of architectural information Update of ASIL as attribute of safety requirements and elements Update of ASIL as attribute of sub-elements of elements Validation plan* Verification plan* Verification specification* Verification report*	see sections above	see sections above

Even if there are lots of work products to consider, we do not have to formulate many requirements for the AMALTHEA meta-model. The reason is that the AMALTHEA meta-model is already well-structured. Especially the mapping and software models allow us to trace important information. As usual, we need to add some properties to requirements.

Table 2.21: AMALTHEA Requirements: Software Safety Requirements Specification

ID	Ref.	Description
SWR.1	6.6.4.1	Software safety requirements shall be designated as such.
SWR.2	6.6.4.1 6.6.4.2	Software safety requirements shall be derived from the technical safety requirements allocated to software.
SWR.3	6.6.4.1 6.6.4.2	There shall be the possibility to link software safety requirements to the technical safety requirements, from which they are deduced.

Table 2.21: AMALTHEA Requirements: Software Safety Requirements Specification

ID	Ref.	Description
SWR.4	6.6.4.2	Software safety requirements must have the capacity to express timing constraints.
SWR.5	6.7.4.9	Software safety requirements shall be allocated to the corresponding software components.

Notes

- SWR.4: Applies if software safety requirements are represented in a formal language.
- SWR.5: This implies that, e.g. tasks, runnables, labels or semaphores can be referenced by (software) safety requirements.

Table 2.22: AMALTHEA Requirements: Software Architectural Design Specification

ID	Ref.	Description
SWA.1	6.7.4.5	It shall be possible to express both static and dynamic design aspects of software components.
SWA.2	6.7.4.6	All safety-related software components shall be categorised as newly developed, reused without modifications or reused with modifications.
SWA.3	6.7.4.15	Necessary software safety mechanisms shall be specified at architectural level for ASILs (A), (B), C and D.

Notes

- SWA.1: Static aspects are, e.g. data types, external interfaces, architectural constraints, while dynamic aspects are behaviour, data and control flow of software.
- SWA.2: This is necessary for further verification processes. Requirement 3.6.4.1 of ISO 26262 demands this categorisation at item level (top level).

Table 2.23: AMALTHEA Requirements: Software Unit Design Specification

ID	Ref.	Description
SWU.1	6.8.4.2	The software unit design shall be specified using notations dependent on the ASIL. It shall support at least semi-formal notations to permit ASIL C and D development.
SWU.2	6.8.4.3	The software unit design specification shall describe the functional behaviour and the internal design at the level of detail necessary for implementation.

Table 2.23: AMALTHEA Requirements: Software Unit Design Specification

ID	Ref.	Description
SWU.3	6.8.4.3 6.10.4.1	The structure and dependencies of (safety-related) software shall be traceable.

Notes

- SWU.1: The standard only uses the categorisation *recommended* and *highly recommended* here, but there needs to be some justification if highly recommended methods (such as semi-formal notations) are not used. Thus, AMALTHEA should at least support highly recommended methods.
- SWU.3: ISO 26262 requirement 6.8.4.4 might be considered an implementation issue but it implies that AMALTHEA should enable tracing relations of software components bidirectionally. Only inheritance of safety-relevant attributes between associated software components allows to guarantee safety during the whole development process.

2.2.7. General Management of Safety Requirements

There are significant properties of requirements that are not yet covered by the given requirements. Some are given implicitly, others are only described in clauses of ISO 26262, that have no work products as an output. Not all properties of safety requirements are given here (please also refer to Chapter 1) as we only consider statements with an impact on the structure of the AMALTHEA meta-model.

Table 2.24: AMALTHEA Requirements: Management of Safety Requirements

ID	Ref.	Description
MSR.1	8.6.4.2.1	Safety requirements need to be designated as such.
MSR.2	8.6.4.2.5	Safety requirements shall have the an ASIL attribute.
MSR.3	8.6.4.2	Safety requirements have to be allocated to software, hardware or interfaces.

Notes

- MSR.1-3: It should be possible to distinguish between safety-related and non safety-related requirements. Requirements as they are currently realised in AMALTHEA (via the external tool ProR) already have a status and an ID. Further, it is already possible to define a hierarchical structure. But it must be ensured in verification processes, that information is not duplicated and that the set of requirements is complete and consistent.

3. Relations between the proposed Meta-Model Extensions of AMALTHEA and the SAFE project

The ITEA2 project SAFE (Safe Automotive soFtware architEcture, 2011-2014) [6] aimed to extend the AUTOSAR architectural model to achieve compliance with ISO 26262. Here, we want to investigate, if it is possible to transfer some of the SAFE results to AMALTHEA4public. The approach of SAFE was the analysis of the missing steps towards the compliance of AUTOSAR with ISO 26262, to define a proposal for an AUTOSAR meta-model extension and finally integration and implementation of the results. In the context of AMALTHEA4public we follow a similar process. We thus compare the results of both projects at every step to extract helpful information from SAFE results.

3.1. The SAFE project

SAFE had three main objectives: The first one was to extend the AUTOSAR architectural model in a way, such that artefacts, that are associated with ISO 26262 can be integrated effectively. This extension was supposed to be implemented in a so-called *reference technology platform* (RTP). The second objective was to enhance methods addressing safety goals, requirements and their validation and to integrate these methods within the RTP. The last objective was to define an ISO 26262-compliant process on the top of model-based development using AUTOSAR.

The expected key results were, amongst other things, to define

- a complete SAFE technology platform for development of automotive products according to ISO 26262 and to perform early safety analyses at architectural level,
- proposals for an extension the AUTOSAR standard in form of a system, hardware and software meta model.

All necessary information about the SAFE project, including all deliverables, can be obtained from the project's website [6].

3.2. SAFE Parts in the Scope of AMALTHEA4public

As already mentioned before, one goal of the AMALTHEA4public project is to extend the AMALTHEA meta-model enabling the representation of artefacts and processes associated to ISO 26262. In Chapter 2 we defined some requirements concerning the meta-

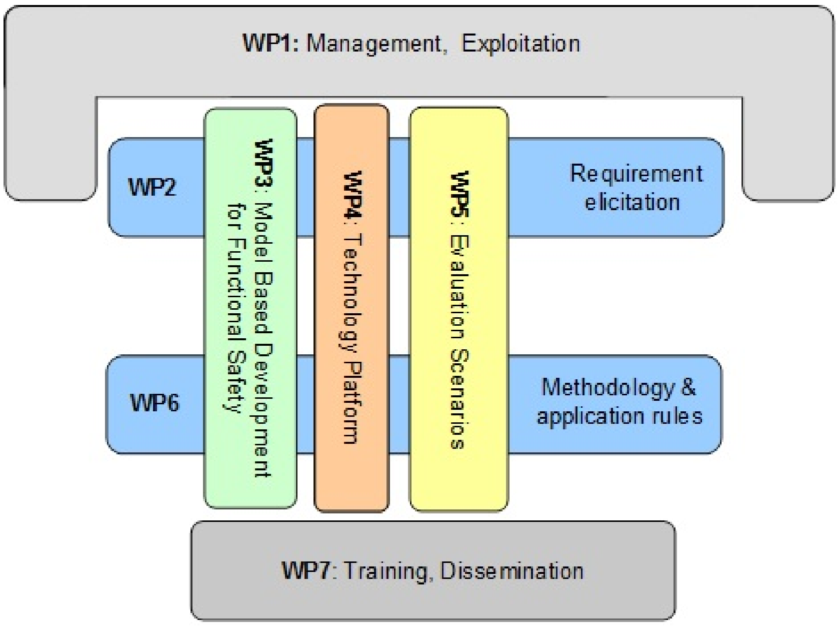
WP1: Management, Exploitation
WP2
Requirement elicitation
WP3: Model Based Development for Functional Safety
WP4: Technology Platform
WP5: Evaluation Scenarios
WP6
Methodology & application rules
WP7: Training, Dissemination

Figure 3.1.: Overview of SAFE Work Packages (source: [11])

model of AMALTHEA. In the remainder of the project we will aim to realise (parts of) these requirements.

Figure 3.1 gives an overview of the structure of the SAFE project. There are seven work packages in SAFE, where content-wise work packages 2 to 6 are the most important. Requirement elicitation and methodology & application rules addressed in work packages 2 and 6, respectively, are interwoven with work packages 3 to 5 concerning model-based development for functional safety, the technology platform and evaluation scenarios to ensure that theoretical approaches can be implemented to the existing model.

In the following, we will focus on WP2 and WP3, where OFFIS was contributing.

In work package 2, ISO 26262 and the state of the art were analysed to derive requirements concerning the new model. One of the results was a table defining about 500 requirements. These requirements consider the meta-model (in form of product artefacts or work products) as well as processes, methods and the infrastructure of the model. References to the corresponding ISO 26262 requirements are given.

The table is not complete with respect to the whole scope of ISO 26262. Only project-relevant requirements are formulated and even some of them are excluded later. Notably, ISO 26262, Part 7 about production and operation is almost not covered by SAFE (the same applies for AMALTHEA). From other parts, some clauses are not regarded. See Figure 3.2 for an overview of the coverage of SAFE with regards to the safety standard. As we will see later, our goals in AMALTHEA4public are quite similar.

The third work package of the SAFE project addresses model based development for functional safety. Based on the results of WP2, proposals for extensions of the meta-

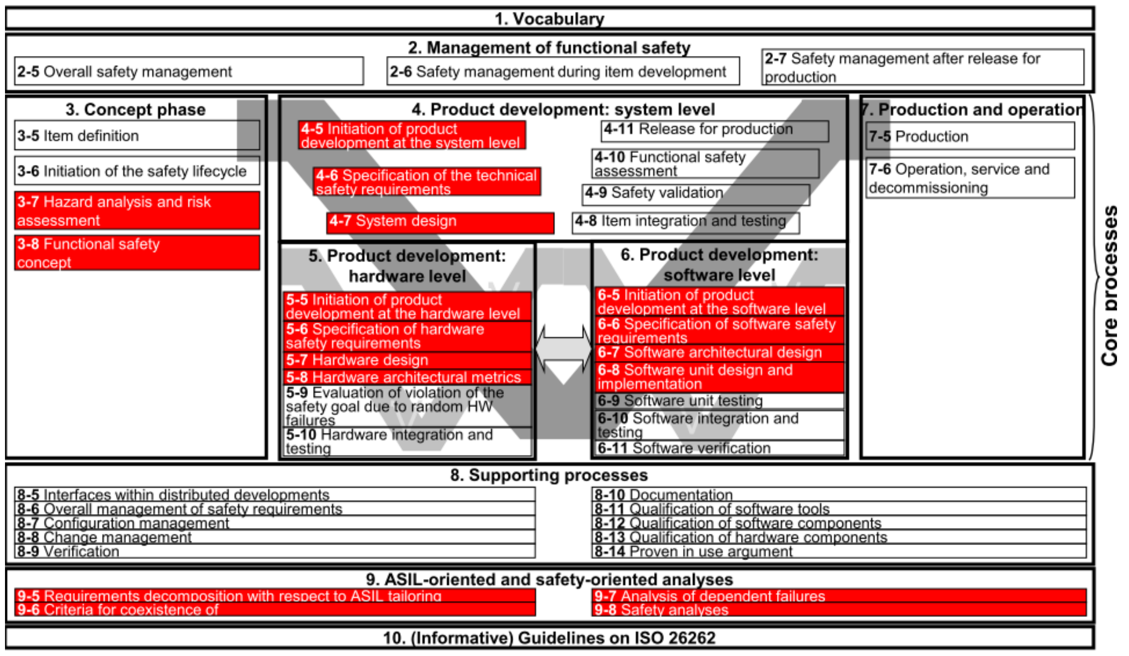
1. Vocabulary
2. Management of functional safety
2-5 Overall safety management
2-6 Safety management during item development
2-7 Safety management after release for production
3. Concept phase
3-5 Item definition
3-6 Initiation of the safety lifecycle
3-7 Hazard analysis and risk assessment
3-8 Functional safety concept
4. Product development: system level
4-5 Initiation of product development at the system level
4-6 Specification of the technical safety requirements
4-7 System design
4-11 Release for production
4-10 Functional safety assessment
4-9 Safety validation
4-8 Item integration and testing
7. Production and operation
7-5 Production
7-6 Operation, service and decommissioning
5. Product development: hardware level
5-5 Initiation of product development at the hardware level
5-6 Specification of hardware safety requirements
5-7 Hardware design
5-8 Hardware architectural metrics
5-9 Evaluation of violation of the safety goal due to random HW failures
5-10 Hardware integration and testing
6. Product development: software level
6-5 Initiation of product development at the software level
6-6 Specification of software safety requirements
6-7 Software architectural design
6-8 Software unit design and implementation
6-9 Software unit testing
6-10 Software integration and testing
6-11 Software verification
Core processes
8. Supporting processes
8-5 Interfaces within distributed developments
8-6 Overall management of safety requirements
8-7 Configuration management
8-8 Change management
8-9 Verification
8-10 Documentation
8-11 Qualification of software tools
8-12 Qualification of software components
8-13 Qualification of hardware components
8-14 Proven in use argument
9. ASIL-oriented and safety-oriented analyses
9-5 Requirements decomposition with respect to ASIL tailoring
9-6 Criteria for coexistence of
9-7 Analysis of dependent failures
9-8 Safety analyses
10. (Informative) Guidelines on ISO 26262

Figure 3.2.: Overview of the Coverage of ISO 26262 in SAFE (source: [11])

model were expressed. They were adapted and refined in the course of the project and examine different topics of ISO 26262. In this context, we consider the following deliverables as the most relevant:

- D3.1.1.b Final proposal for extension of meta-model for hazard and environment modelling [7]
  Proposal on how to integrate the ISO-artefacts item, hazard, risk descriptions, safety goals, safe states, operating modes, requirements and some further safety-relevant concepts.

- D3.1.2.b Final proposal for extension of SAFE meta-model for safety requirement expression modelling [8]
  Describes how modelling of safety requirements is supported by the meta-model. In particular, the model is extended to allow refinement, allocation and traceability of safety requirements

- D3.2.1.b Final proposal for extension of meta-model for software and system modelling [9]
  Focuses on the architecture behind the meta-model, in particular system and software modelling. The integration of all of these concepts (including other results of WP3, that are mentioned here) is given in Deliverable D.3.5.c.

- D3.5.c, D3.5.d Final release of System, SW, HW reference meta-model defini-

tion [10]
State of integration with the platform (WP4), the updated version D3.5.d describes a complete meta-model, which extends AUTOSAR and EAST-ADL.

3.3. Expected Results from the SAFE Project

As the scope of SAFE has a significant overlap with AMALTHEA4public, we can analyse our requirements with respect to the SAFE requirements. Focusing on the SAFE requirements that deal with product artefacts and work products, we already identified matches.

To be more exact, we can even track all our requirements to SAFE requirements, even if for some aspects the emphasis in SAFE is different. For example, we will not address hardware development in AMALTHEA4public, but SAFE defined a lot of requirements concerning this subject.

Later in the project we expect to extend the data model of AMALTHEA in an ISO 26262 compliant way. To this end, we will further analyse the parts of the SAFE meta-model, that cover gaps that we identified in this document. Potentially, SAFE results can be adopted within AMALTHEA4public.

4. Conclusion

We have identified gaps in the AMALTHEA meta-model in its current state with regards to the automotive safety standard ISO 26262 and formulated them as requirements for AMALTHEA. The analysis focused on the work products defined in ISO 26262 and their associated requirements. Requirements in ISO 26262 that were outside of the current application scope of AMALTHEA were excluded.

While many work products defined in ISO 26262 are already sufficiently represented in AMALTHEA, there is the potential for significant improvement with regards to the coverage of the standard. Solutions to close the identified gaps were not yet proposed in this document and will be considered in a next step – also in collaboration with the other work packages in AMALTHEA4public. The goal is to improve the coverage of the AMALTHEA platform for safety-relevant developments conformant to ISO 26262.

A. Appendix

A.1. ISO 26262 Work Products Excluded from Gap Analysis

The following work products are excluded from the gap analysis. They are deemed out of scope for the current applications of the AMALTHEA platform and thus do not need to be represented in the meta-model.

Part	**Work Product**
2.5	Organisation-specific rules and processes for functional safety
2.5	Evidence of competence
2.5	Evidence of quality management
2.6	Confirmation measures reports
2.6	Functional safety assessment plan
2.6	Project plan
2.6, 6.5, 8.14	Safety plan
2.6	Safety case
2.7	Evidence of field monitoring
4.8	Item integration and testing plan
4.8	Integration testing specification(s)
4.8	Integration testing report(s)
4.9	Validation report
4.11	Release for production report
5.7	Hardware design verification report
5.7	Hardware integration and testing report
5.7	Hardware safety analysis report
5.9	Analysis of safety goal violations due to random hardware failures
5.8	Analysis of the effectiveness of the architecture of the item to cope with the random hardware failures
5.9	Review report of evaluation of safety goal violations due to random hardware failures
5.9	Review report of evaluation of the effectiveness of the architecture of the item to cope with the random hardware failures
5.9	Specification of dedicated measures for hardware
6.5	Design and coding guidelines for modelling and programming languages
6.5	Tool application guidelines
6.10	Embedded software
6.C	Calibration data
6.C	Calibration data specification

6.C	Configuration data
6.C	Configuration data specification
7.5	Assessment report for capability of the production process
7.5	Control measures report
7.5	Safety-related content of the production control plan
7.5	Safety-related content of the production plan
7.5	Specification of requirements on the producibility at system, hardware or software development level
7.6	Repair instructions
7.6	Safety-related content of the maintenance plan
7.6	Safety-related content of the information made available to the user
7.6	Instructions regarding field observations
7.6	Safety-related content of the instructions for decommissioning
7.6	Specification of requirements relating to operation, service and decommissioning at system, hardware or software development level
8.5	Development interface agreement (DIA)
8.5	Functional safety assessment report
8.5	Supplier selection report
8.5	Supplier's project plan
8.5	Supplier's safety plan
8.5	Supply agreement
8.7	Configuration management plan
8.10	Documentation management plan
8.10	Documentation guideline requirements
8.11	Software tool criteria evaluation report
8.11	Software tool qualification report
8.13	Qualification plan
8.13	Hardware component test plan
8.13	Qualification report
8.14	Description of candidate for proven in use argument
8.14	Proven in use analysis reports
9.8	Safety Analyses

Bibliography

[1] Eclipse EMF: *Eclipse Modeling Framework.* – URL `http://www.eclipse.org/modeling/emf/`

[2] Eclipse RMF: *ProR - A Tool for working with Requirements.* – URL `http://www.eclipse.org/rmf/pror/`

[3] ISO: *ISO 26262 - Road vehicles — Functional safety.* Juli 2009

[4] ISO: *ISO 26262 - Road vehicles — Functional safety — Part 2 Management of functional safety.* Juli 2009

[5] ISO: *ISO 26262 - Road vehicles — Functional safety — Part 3 Concept phase.* Juli 2009

[6] SAFE: *Safe Automotive soFtware architEcture.* 2011-2014. – URL `http://safe-project.eu`

[7] SAFE: *Deliverable D3.1.1 b: Final proposal for extension of meta-model for hazard and environment modeling.* März 2013

[8] SAFE: *Deliverable D3.1.2b: Final proposal for extension of SAFE meta model for safety requirement expression modeling.* März 2013

[9] SAFE: *Deliverable D3.2.1.b: Final proposal for extension of meta-model for software and system modeling.* Februar 2014

[10] SAFE: *Deliverable D3.5.c: Final release of System, SW, HW reference meta-model definition.* Februar 2014

[11] SAFE: *Safe Automotive soFtware architEcture (SAFE) Project Presentation*