(ITEA 3 – 17003)

PANORAMA
Boosting Design Efficiency for Heterogeneous^3 Systems

Deliverable: D6.5
Design Handbook

Work Package: 6
Design Flow and Traceability

Task: T6.5
Integrated Safety Analysis

Document Type:	Deliverable	**Classification:**	Public
Document Version:	Final	**Contract Start Date:**	2019-04-01
Document Preparation Date:	2022-03-31	**Duration:**	2022-03-31

ITEA 3
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

EUREKA

Authors

Karsten Albers Inchron GmbH

Mahmoud Bazzal Dortmund University of Applied Sciences and Arts

Jan Steffen Becker OFFIS e.V.

Maria Bonner Siemens AG

Daniel Fruhner Dortmund University of Applied Sciences and Arts

Björn Koopmann OFFIS e.V.

Olexiy Kupriyanov Siemens AG

Harald Mackamul Robert Bosch GmbH

Philip Okonkwo Dortmund University of Applied Sciences and Arts

David Schmelter Fraunhofer IEM

Jan-Philipp Steghöfer University of Gothenburg

Ingo Stierand OFFIS e.V.

Marc Zeller Siemens AG

Contents

List of Figures

List of Tables

Summary

This document describes a design process for the development of heterogeneous, safety-critical systems as considered in the PANORAMA project. The process has been defined as a flow of activities and the work products that provide the interfaces between activities. It is divided into three phase, namely *System Analysis*, *System Safety Design*, and *Assessment and Optimization*, to which the activities are assigned.

Many aspects must be considered beyond a simple structural view for instantiating such a design process. The document gives an overview of the aspects that played an important role in the development of the process, among which are collaboration and requirements imposed by existing safety standards. Particularly, the safety aspect is an important driving factor for the development of the process. The activities are aligned with the ISO 26262 standard, and a coverage analysis with the corresponding mandatory clauses in the ISO standard is performed.

The document discusses a total of 12 engineering scenarios, which have been implemented by the project partners and fit into the process structure. They exemplify the instantiation of parts of the process and thus contribute to demonstrating its applicability. Finally, an analysis is performed on the results in order to assess the actual coverage of the process with respect to the engineering scenarios as well as the completeness and consistency of the interfaces between these scenarios. Finally, a gap analysis is conducted in order to identify open challenges and to set the stage for future activities.

1 Introduction

Software developers in the automotive and avionics industries are exposed to foundational changes in how systems are architected and developed. Most importantly, electrical architectures in vehicles and aircraft have become highly networked and increasingly include specialized and networked hardware, such as AI accelerators for deep learning in autonomous vehicles. Centralized hardware platforms are complemented with specialized hardware and software for functions such as drive, comfort, and multimedia. At the same time, collaboration between partners along the value chain becomes more important as software providers, domain experts, and service providers will become more involved in future development processes.

The increased heterogeneity on all these levels may, however, not impede the safety of the system. On the contrary ensuring safety becomes even more important in systems that include learning components and that make autonomous decisions in situations that are impossible to fully predict at run-time [GJW+20]. Safety standards are adopted in all relevant domains (e.g., ISO 26262 [Int18] for automotive, ISO 17894 [Int05] for maritime, and ARP 4754/61 [Soc96] for avionics).

In the light of these changes and challenges, the automotive and avionics industries are changing their methods, tools, platforms, and their *development processes* (see, e.g., [SKHW19; HWS17]). Collaboration during development and close cooperation between OEMs and suppliers will become the norm. An agile mindset, which has transformed OEMs in the past years, is increasingly applied to suppliers as well [HMSS16]. Instead of treating suppliers and the components they deliver as black boxes, they become increasingly integrated into the development process and requirements and systems adapt during development. Suppliers thus work more closely with the product teams, but need to show more flexibility as they track the product development. Continuous safety engineering also requires them to continuously provide analysis result and participate in the ongoing safety argumentation (see, e.g., [BL21]).

1.1 Scope and Objectives

This *design handbook* intends to showcase how the results of the PANORAMA project can be integrated into a development process that supports the design of safety-critical systems in the light of the challenges outlined above. To address this emerging new development paradigm, this report answers the following research questions:

RQ-1 What must a system engineering process look like to enable the standard-compliant development of heterogeneous systems?

RQ-2 What are typical engineering scenarios? How does the process support these scenarios?

RQ-3 Which tools can be used to implement the process? How can tool interoperability be ensured along the process?

RQ-4 How can applicability of the process be ensured?

We have formulated these research questions based on the needs of a number of partners from the avionics and automotive industry who participated in PANORAMA. These organizations are working towards integrating suppliers and OEMs more closely in the development processes, increasing the complexity of the systems under construction, reducing the time to market, and delivering safe and reliable products.

By answering these questions, we provide the following contributions:

Design Process We investigate which activities and work products are relevant for such a process and which commonalities the different standards prescribe (cf. Chapter 4).

Engineering Scenarios Engineering scenarios describe typical sub-processes engineers of safety-critical systems conduct. We identify such scenarios and show how the process we designed supports them (cf. Chapter 5).

Tool Support We identify commonly used tools that support the different activities of the design process and show how standardized interchange formats can be used to enable exchange of information between the relevant stakeholders in the process (cf. Chapter 6).

Analysis of Engineering Scenarios We analyze the engineering scenarios in terms of which aspects of the relevant standards they cover, which patterns they include that can be reused in other contexts, etc. Overall, this provides insights into the applicability of the design process and the generalizability of our findings (cf. Chapter 6).

1.2 Outline

The document is structured as follows: In Chapter 2, we introduce the relevant background, including safety standards and collaboration approaches. We describe the methodology we followed to arrive at our results in Chapter 3. Based on this, we showcase the design process in Chapter 4 and detail the activities, work products, tools, and domain-specific languages that are relevant for it. Engineering scenarios are then introduced in Chapter 5 before their analysis is discussed in the context of the design process in Chapter 6. Chapter 7 concludes the report.

2 Background

The definition of an applicable design process, with its phases and steps, their interfaces and (intermediate) work products depends on numerous factors, such as effective regulations for the intended work products, infrastructure and organizational structure, as well as existing tools, best practices and legacy support. One of the main objectives of the PANORAMA project is to establish a design flow that:

- enables covering the engineering challenges the project is aiming at in industry-relevant setups (cf. Chapter 1); and that
- can be instantiated with the methods and tools developed in the project for a representative set of engineering scenarios.

To this end, we documented the relevant state of practice as well as requirements from the industrial partners for such a process in Deliverable 6.1 [MCA+20]. The findings of this deliverable are summarized in Section 2.1. In addition, we discuss the role of different safety standards on the definition of the design process in Section 2.2 and the influence on collaboration between different organizations in Section 2.3.

2.1 Existing Design Processes as described in Deliverable 6.1

We collected the *state-of-the-art of collaborative development processes* by conducting a literature review and a review of results from previous projects, in particular projects with strong industrial focus such as AMALTHEA4public, ARAMiS II, and DEIS. In AMALTHEA4public, e.g., first attempts have been made to map out a generic design process and show which activities and tools can be used to achieve certain objectives defined in ISO 26262 [TMSP16]. We also considered how collaboration can be achieved from a tooling perspective, e.g., by considering version control systems and real-time collaboration tools.

One of the outcomes of the ARAMiS II project was a design process that is designed to be applicable in many domains and aligned with different safety standards. Likewise, Siemens has already described a collaboration workflow for use with their Polarion platform. These processes are not complete in the sense that they cover the entire development life-cycle. For instance, timing and safety analysis activities are not featured prominently in either.

One important focus of both the AMALTHEA4public and the DEIS project was interchange of information. In AMALTHEA4public, the AMALTHEA model has been defined as a way to exchange information about system design, timing, hardware, and other aspects. In DEIS, the Open Dependability Exchange (ODE) model has been defined as a way to exchange information about safety and reliability concerns. Both models have been taken up and refined in PANORAMA and form an important foundation of information exchange. Any collaborative design process needs to ensure that information can be exchanged freely and can be understood, analyzed, and created by the involved parties.

To define the *state-of-practice of collaborative development processes*, we collected use case descriptions from the industrial partners that contained typical collaboration scenarios during a typical endeavor, ran an ideation workshop to collect ideas and requirements for improved collaboration, and held a focus group to understand these scenarios and their challenges better. The results are structured along different dimensions of the daily practice:

Collaboration Workflows change during the project duration and differ in intensity according to the current phase. Depending on the relationship between partners, different contractual solutions are used and long-term relationships between partners can decrease formality of the arrangements.

Artifact Exchange is of vital importance for the success of the endeavor. The level of formality of the artifacts changes over the course of the endeavor with less formal requirements exchanged in the beginning and more formal and structured artifacts such as Matlab/Simulink models exchanged later on. In any case, it is important that file formats are standardized and tool providers are included in the process of defining standards for the artifacts.

Infrastructure to process these artifacts also need to be set up and integrated into other tools such as application life-cycle management tools or issue tracking tools.

Traceability between the individual work artifacts needs to be established, especially if they are updated frequently by different parties. Clearly established and maintained traceability links support, among other things, change impact analysis and allow for the easy and (semi-)automatic creation of reports that are required for certification purposes.

Security and Intellectual Property Management needs to be taken into consideration whenever IP needs to be protected. When different partners collaborate and exchange information, some of it might be sensitive and should, e.g., be prevented from introspection by unauthorized parties.

Based on these findings, we established a number of *requirements for future collaborative development processes*. The two main sources of requirements are the ISO 26262 safety standard used for the development of vehicles and the ideation workshop mentioned above. While the full list of requirements can be found in [MCA+20], it is worth noting that these two sources provided requirements on the process itself, the models that are exchanged,the verification and validation tools that are used, and the way knowledge is managed.

The final step described in Deliverable 6.1 is a *gap analysis* between these requirements and state of the art and practice. Noteworthy gaps are:

- there is a need to create and maintain safety cases in form of models;
- there is a need to capture design decisions explicitly in the model itself;
- there is a need for a standardized tool environment with tools flexible enough to support different concrete projects; with standardized exchange formats for the most common artifacts; and with extensibility to address customer-specific requests;
- there is a need to use trace information for program comprehension or to maintain an overview of the system;
- there is a need to link analysis results to quality attributes that can be improved in the model;
- there is a need to have a standard process in-house and have a way to synchronize this standard process with each supplier;

PANORAMA provide solutions for a number of these gaps. Not all of these solutions are addressed in this deliverable, but deliverables from other work packages cover, e.g., the creation of safety analysis artifacts needed for model-based safety cases. This deliverable will focus on process aspects and, in particular, will focus on how a generic process description supports a number of engineering scenarios based on standardized tooling and common exchange formats.

2.2 The Role of ISO 26262 in the Definition of the Design Process

For systems where functional correctness and safety is of high importance, there exist standards such as IEC 61508 [Int98], which provide guidance on how to address the safety-related aspects in the development process such as the assessment of risks, derivation of safety requirements and adequate design steps to assure that the developed system fulfills these requirements. It was adopted in different domain specific standards like ISO 26262 (automotive) [Int18], ISO 17894 (maritime) [Int05] and ARP 4754/61 (avionics) [Soc96].

Although each domain-specific safety standard defines a specific vocabulary, it covers the complete safety life-cycle derived from the IEC 61508 [Int98] safety standard. First attempts to identify similarities and dissimilarities of different safety standards have already been performed in [BBBD+10; BABB+12; LABB+12; MBBB+12; PA99]. As a result of these previous analyses the following similarities have be identified:

- Common notion of safety and certification
- Linear progressing safety process with dedicated phases
- Combined hazard assessment and risk analysis to derive safety requirements
- Criticality levels as means to allocate safety (integrity) requirements to system elements

- Verification activities are driven by the safety requirements
- Safety case provides evidence that safety requirements are fulfilled which is needed for certification

Moreover, the following divergences have be identified:

- Varying definition of criticality levels
- Different approaches for the allocation of safety requirements
- Domain-specific verification & validation processes

Based on a number of identified similarities of the safety standards in the transportation sector, [PA99] already outlines a generic safety assessment process integrated into a concrete system development process.

In this document we use the ISO 26262 process as an example for a safety life-cycle, which is based on the well established *V-model* development life-cycle. However, the results can be transferred to any other safety standard derived from the IEC 61508 safety standard, due to the fact the the safety life-cycle is the same.

2.3 The Role of Collaborative Engineering in the Definition of the Design Process

As explained in the introduction, we see a trend firstly toward integrated, heterogeneous functional domains in the automotive and aerospace industries and secondly towards heterogeneous hardware architectures based on capable, centralized hardware platforms. At the level of development processes and collaborations along value chains, these trends imply two things: (1) OEMs and suppliers at several levels of the supplier pyramid must collaborate more closely and intensively than before for successful function integration. (2) In order to make the complexity of the developed software-intensive products manageable, the partners involved depend on a simple and efficient exchange of different development artifacts, such as enabled by AMALTHEA system models.

However, in such heterogeneous communities, typically suppliers collaborate whose business areas overlap. Thus, individual companies are in competition with each other and an open exchange of information is not readily possible due to intellectual property protection concerns.

For this reason, we investigated in PANORAMA what organizational challenges and threats arise for data security in heterogeneous development communities when sharing development artifacts, especially detailed AMALTHEA system models. Based on this analysis, we have conceived recommendations on how to enable a trustful exchange of such development artifacts. Hereto, we conducted a total of four workshops over a period of six months in Q4 2020 and Q1 2021 with partners of the PANORAMA project consortium. We conducted each workshop in an agile manner and academic partners supported their execution methodologically. This means that at the beginning of each workshop, we conceived the respective workshop topics, prioritized them and worked on them jointly. At the end of each workshop, we conducted a review of the workshop results and a retrospective of our joint collaboration in the workshop. This ensured that we were able to work efficiently towards the defined research questions and designed subsequent workshops based on the gained insights. In the course of the workshop series, we developed a fictional, minimal but sufficient and representative engineering scenario. In this scenario, four companies collaboratively develop an advanced driver-assistance system (ADAS) across three stages of the value chain. To analyze the cross-company information exchange, we instantiated the PANORAMA design process multiple times, i.e., per company.

We modeled the identified data security threats by means of the existing threat modeling methods LINDDUN [DWS+10] and STRIDE [HL06]. Our recommendations for data security are based on the international standard ISO/IEC 27010 [ISO15].

We describe our research methodology on data security in collaborative processes and our results in more detail in Deliverable 6.3 [SSK+21].

3 Methodology

In order to answer our research questions (cf. Section 1.1), we followed a multi-stage research methodology that involved a number of researchers and industrial and academic partners over a period of several months. Roughly, the methodology incorporated the following steps:

Focus Group to come up with a process description based on typical industrial workflows.

A Second Focus Group to refine the process with aspects of collaboration and traceability.

A Survey of Tools and Artifacts typically used in industry and how they are related.

Continuous Refinement over several months by experts to incorporate additional aspects, in particular safety.

Creation of Engineering Scenarios to identify which concrete activities are commonly performed.

Systematic Analysis of Engineering Scenarios to analyze coverage and identify gaps.

We describe these distinct steps in our methodology in the following.

3.1 Focus Group for Process Definition

The first step towards a process description took place as a focus group in which three engineers from three companies were assisted by two researchers in creating a first version of a "big picture" development process description for heterogeneous systems. The main focus of this exercise was to create a description that was independent of a specific safety standard and showed different analysis techniques that would be applied based on a common system model, the AMALTHEA model [HMS+17]. As such, the process description was not complete at this stage and was very high-level. However, it showed a complete life-cycle from requirements engineering, via system design, to system analysis. The analysis results fed back into the system design stage. While the consensus of the engineers was that the process overview is compatible with common standards in the automotive and the avionics domain, no attempt was made to map the activities to these standards.

3.2 Focus Group on Collaboration

We conducted a second focus group with seven engineers from five different companies. These companies included an OEM from the avionics industry, a tier-1 supplier (mostly automotive industry), and three tool suppliers. The aim of the focus group was to understand how companies collaborate in a large product development effort, which development activities are conducted jointly and separately, how information is exchanged, and which traceability is established and maintained. In addition, we wanted to better understand how the process description developed in the first focus group fits into a collaborative work environment.

The focus group was prepared by the researchers by providing an interview guide. Two researchers took notes while two other researchers asked the questions, making sure all participants were provided an opportunity to answer every question. The full focus group was recorded and two transcripts were created independently. No major differences in these transcripts were detected.

The transcribed focus group then served as the input to an analysis activity which resulted in the definition of requirements and gaps (cf. Deliverable 6.1 [MCA+20]), but also in a refinement of the process description. We used thematic coding and a workshop among the involved researchers in this process.

3.3 Survey

To get an understanding of which tools are used, which artifacts these tools produce and consume, and which activities these tools support, we sent out a survey to eight tool providers. The tool providers (three academic, five industrial) were asked to answer the following questions:

- What are the model elements (artifacts) consumed/produced by the tool?
- What is the format of each model artifact?
- What are the design steps (activities) where the tool shall be used?

The results were collected in a table that listed the various artifacts and which tools consume/produce them. In addition, a list of activities was synthesized from the answers and matched to the process description, leading to a refinement of it. The tools we collected were also used to extend the process description.

3.4 Continuous Refinement

After the initial data collection steps, we worked on the continuous refinement of the process description. Importantly, the development of the MobSTr dataset [SKB+21] provided crucial insight into missing activities, in particular w.r.t. safety activities. All refinements were suggested and discussed by a group of researchers and evaluated with industrial partners that also participated in the initial steps.

The end result of the refinement is a process description that contains all crucial process steps in the development of a heterogeneous system. It remains independent of a concrete safety standard, even though we did perform a mapping to ISO 26262 [Int18] as part of the refinement as presented later in this report.

3.5 Creation of Engineering Scenarios

In order to understand how the process description supports common tasks in the engineering of heterogeneous systems, we asked our academic and industrial partners to provide *engineering scenarios*, i.e., standardized descriptions of such common tasks. For this purpose, we created a template that could be filled out by the project partners (cf. Table 3.1). We had pre-filled some of the information for some partners based on other information available to use in the project.

We received twelve engineering scenarios from nine different partners. Since our partners are mainly involved in system analysis, the scenarios are focused on these aspects of the process. The resulting engineering scenarios were discussed with the partners providing tools in order to ensure that they match the tool provider's own view on usage of their tool and methods.

Each Engineering Scenario is presented as an extended *user story* [LDvdWB16]. In general, a user story asks for a *persona* (who is responsible?), a *job to be done* (what is the task to be solved?), the *pain* (what is the problem?), a *pain reliever* (how is the problem solved?), and the *benefit* (from solving the problem). The template is shown in Table 3.1.

Table 3.1: Template for the Collection of the Engineering Scenarios

ES-n	Title: *tbd.*
	Provider(s): *tbd.*
Goal – *What is the goal?* This section describes the purpose of the PANORAMA tool or method shown in the engineering scenario. It describes on a high level the *job to be done*, the *pain*, the *pain reliever* (i.e., the tool or method), and the *benefit.*	
Activities in Focus – *Which activities are in focus?* This section describes the *job to be done* and the *pain reliever* by showing the single activities that make up the engineering scenario and the data flow between them. In particular, it is shown which parts of the *PANORAMA Design Process* (see Chapter 4) are supported by the tool/method in the focus of the scenario.	
Input Artifacts – *Which artifacts are consumed?* This section summarizes the input artifacts of the tools used in the scenario. The information on artifacts and their formats is key for forming interoperable tool chains that support the process.	**Output Artifacts** – *Which artifacts are produced?* This section summarizes the output artifacts of the tools used in the scenario. The information on artifacts and their formats is key for forming interoperable tool chains that support the process.

Traceability Information – *Which (cross-model) trace links are used?* As identified in the second scenario and also in general in WP6, traceability strongly supports the creation of safety cases. As part of the MobSTr dataset, a traceability information model (TIM) that supports the safety process has already been identified. Since not all tools within PANORAMA have been integrated into MobSTr or provide other traceability information, this section is optional.
Example Instantiation – *How can the scenario be applied?* This section provides an example for application of the tool or method in the context of the engineering scenario. Because not all providers of Engineering Scenarios participated directly in WP6, example instantiations are missing for some of the Engineering Scenarios. The interested reader is referred to the WP3 deliverables and tool presentations for examples in this case.

3.6 Analysis of Engineering Scenarios

Once we received the scenarios, we defined different criteria for their analysis. As they are mostly concerned with analysis tasks, we also consider the different activities we conducted in the creation of the MobSTr dataset [SKB+21] as an additional data point. This, in particular, provides additional input for requirements and system design phases which are not well-covered by the engineering scenarios we collected.

As a first step, the collected scenarios were analyzed with regard to their coverage of the design process. For this purpose, the process steps addressed by the engineering scenarios were combined into a common overview. The resulting overview was then used to determine coverage metrics on process level and from the perspective of parts and clauses of the particularly relevant ISO 26262. Based on these results, progress was assessed in comparison to the outcomes of the AMALTHEA4public project.

A second step was a detailed investigation of the (technical) interfaces between the scenarios considered. Here, a special focus was placed on the consideration of the exchange formats and data models used. In the further course of the analysis, commonly used and non-agreed formats were collected separately to identify gaps in the tool chains.

4 Design Process

4.1 Approach and Overview

A common design process on which all involved engineers and stakeholders can agree is a key enabler for joint systems development by collaborating heterogeneous organizations. Such a process defines the individual design steps and activities, as well as the required input work products and provided output work products for these steps. A work product can be any design artifact like a document, a model, executable code, or analysis results.

The flow of activities and work products provides a general structure, which must be further substantiated. This includes, among many other things, the roles and responsibilities for the individual activities, suitable quality metrics for the work products, and measures to ensure adherence to these metrics (quality gates). An instantiation also requires a proper definition of the interfaces, such as format and structure of the work products. This in turn provides the basis for establishing traceability between individual design artifacts, which constitutes another important pillar in collaborative system development. All this is strongly entangled with the tools that are used for the development.

Figure 4.1 depicts a high-level overview of the design process that has been developed in the PANORAMA project. The main objectives of this development can be summarized as follows:

- Definition of a process that covers as many requirements of safety oriented system development as possible;
- Enabling the investigation of aspects of the development as discussed in the previous sections in the context of a research project; and
- Demonstrating the application of collaborative development by instantiating a set of engineering scenarios, including the application of particular tools and the production and exchange of concrete design artifacts.

The resulting process is consistent with the ISO 26262 standard process: The left hand side of Figure 4.1 shows the main design phases *Concept*, *System Level* and *HW/SW Level* of the standard. The process also consists of three phases, which are however arranged slightly differently. The *System Analysis* phase subsumes the activities concerning early design steps such as hazard and risk assessment, the definition of system items and elements, and the top-level safety requirements. The labels (dark blue) attached to the activities refer to the corresponding clauses in the ISO 26262 standard. For this first phase, the corresponding clauses belong to the *Concept* phase (Part 3).

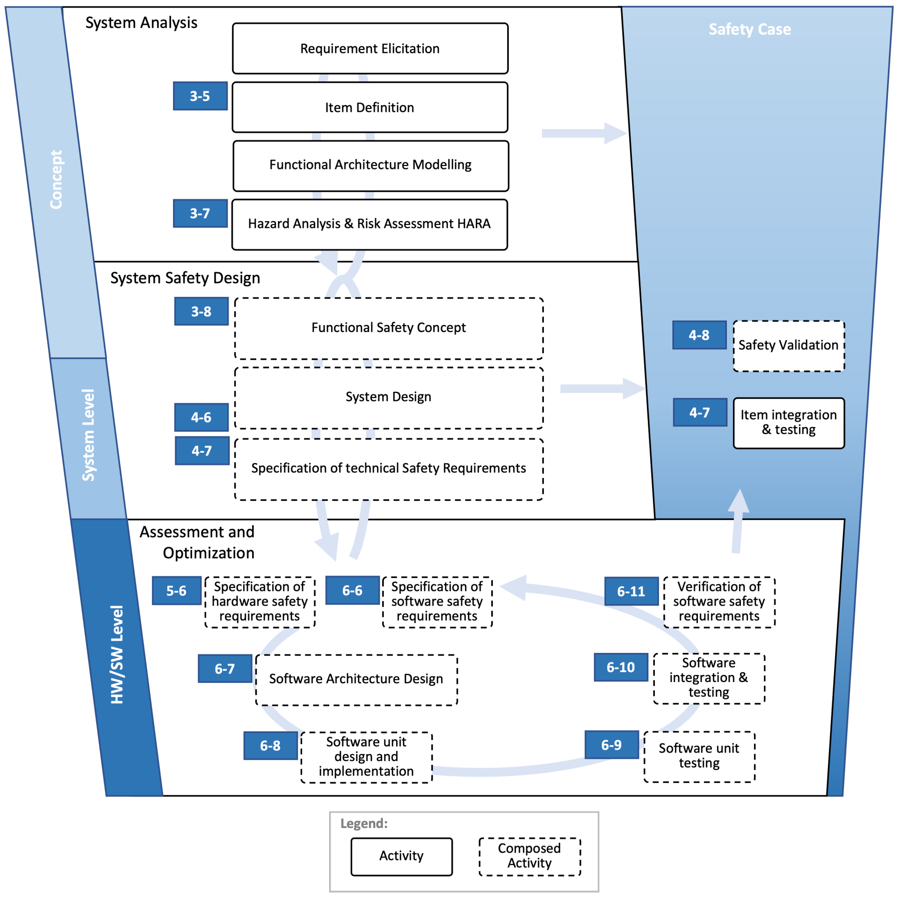

System Analysis
Requirement Elicitation
3-5
Item Definition
Functional Architecture Modelling
3-7
Hazard Analysis & Risk Assessment HARA
Concept
Safety Case
System Safety Design
3-8
Functional Safety Concept
System Design
4-6
4-7
Specification of technical Safety Requirements
System Level
4-8
Safety Validation
4-7
Item integration & testing
Assessment and Optimization
HW/SW Level
5-6
Specification of hardware safety requirements
6-6
Specification of software safety requirements
6-11
Verification of software safety requirements
6-7
Software Architecture Design
6-10
Software integration & testing
6-8
Software unit design and implementation
6-9
Software unit testing
Legend:
Activity
Composed Activity

Figure 4.1: High-Level Overview of the Design Process

Phase *System Safety Design* concerns the construction of the functional safety architecture and its decomposition with respect to the hardware/software interfaces. The former corresponds to Clause 3-8 in the ISO 26262 standard, which is located also in the *Concept* phase. As the presented process strongly focuses on model-based design, the engineering activities for constructing the functional safety architecture are supposed to be performed on a functional architecture model and thus have been located in the system design phase.

The specification of technical requirements is the last activity and its output provides the interface to the *Assessment and Optimization* phase. It is devoted to the design of the technical realization as well as its assessment and optimization. The ISO 26262 standard separates HW and SW level design (Part 5 and 6, respectively). Subsuming them into a single phase reflects the tight connection of these two parts.

The figure reveals that the specified design phases are mostly concerned with the design (left part of the "V") and not with system integration. This is because the project activities are devoted mainly to these phases. Also development on demonstrators that involve system integration took place (cf. [Con22]), which is however out of scope of this handbook. The process nonetheless covers various validation and verification activities.

A particular focus has been put on the construction of safety cases from the individual activities as they constitute a key design artifact in safety oriented design.

4.2 Design Phases

In the following, the phases are examined in more detail to shed some light on the involved development activities. Many relevant engineering aspects have been covered in different engineering scenarios. These instantiate this process and are elaborated in Chapter 5. The following discussion relates the individual activities in the process with the corresponding clauses of the ISO 26262 standard in order to give some guidance for readers who are familiar with the standard. Each phase lists some tools that may be used to support the individual activities. Many of them are rephrased from Trei et al. [TMSP16] for additional guidance. Trei et al. also served as input for this report in other aspects and thus is highly recommended related work. Again, Chapter 5 provides deeper insight into potential tools for instantiating the design process.

4.2.1 System Analysis

The activities and resulting artifacts of the *System Analysis* phase are depicted in Figure 4.2. The activity *Requirement Elicitation* subsumes all steps that define the functional system requirements. They define what the system is supposed to do. The next step is definition of the system items in the *Item Definition* activity. It corresponds to Clause 3-5 of the ISO standard. There, it is recommended that functional and non-functional requirements shall be available. Hence, this activity is preceded by *Requirement Elicitation*, which is not explicitly mentioned in the ISO standard. Also not mentioned is the activity that follows, i.e., the decomposition of the system items into an initial functional architecture in the *Functional Architecture Modeling* activity. A functional architecture has been identified as a suitable structured representation of the system items (cf. [PHAB12]). It may incorporate the results of an *Assessment of External Systems*, which is otherwise mentioned in the ISO standard as part of the *Initiation of the Safety Lifecycle* (Clause 3-6).

The following activities concern the early risk assessment, from which the top-level safety requirements are derived. The results of the *Hazard Analysis and Risk Assessment* serve as inputs for the *Formulation of Safety Goals.* Since timing requirements are considered an important part of system design, they have been made explicit in this picture as *Formulation of Timing Requirements.*

Tools: The following list is purely informative, in no way complete, and not intended as a recommendation. Each item contains the tool name, followed by the artifact format in parentheses, and the activities/work products.

- APP4MC (AMALTHEA): Requirements, Item Definition, Architecture
- Enterprise Architect (SysML): Requirements, Item Definition, Architecture

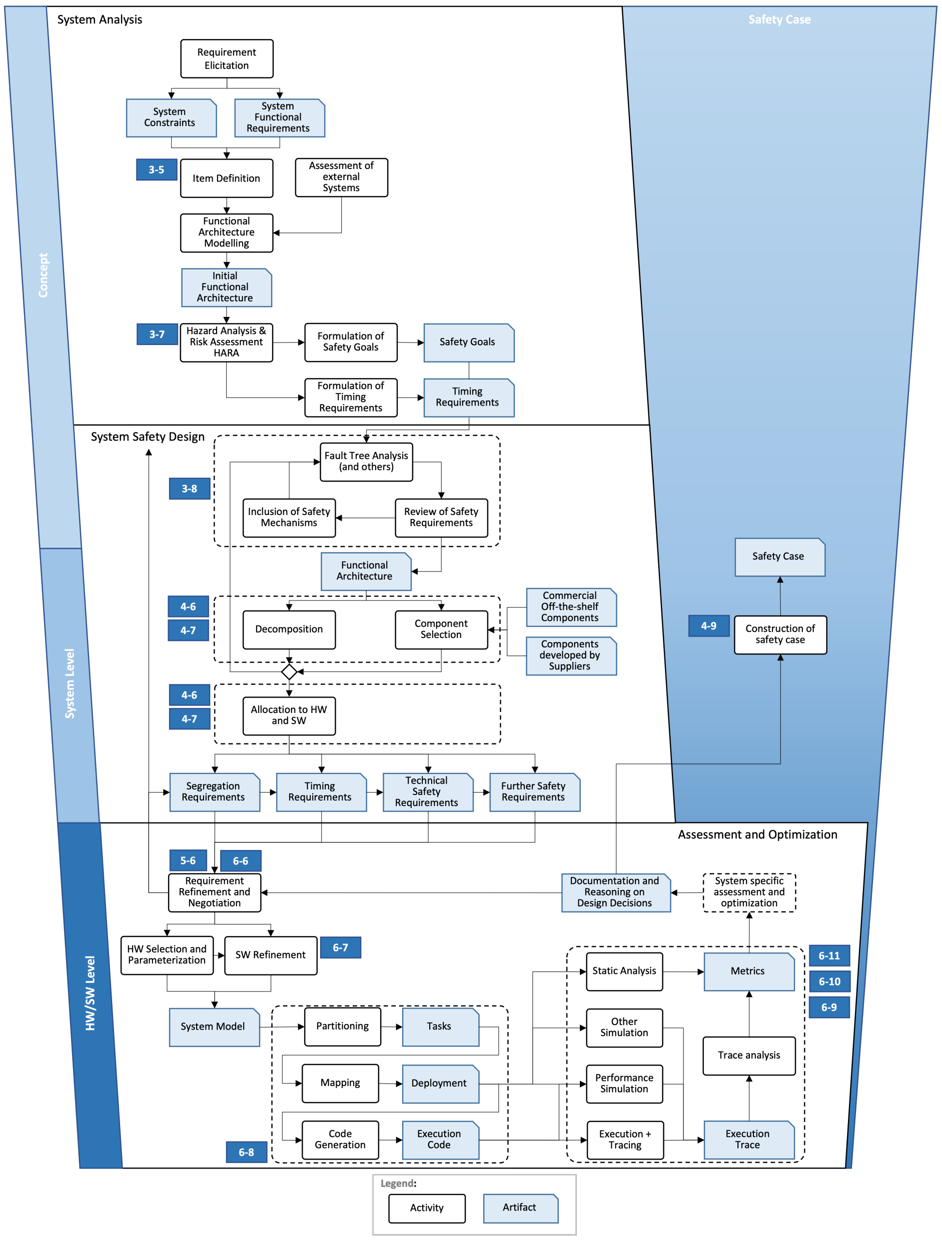

System Analysis
Safety Case
Concept
System Level
HW/SW Level
Requirement Elicitation
System Constraints
System Functional Requirements
3-5
Item Definition
Assessment of external Systems
Functional Architecture Modelling
Initial Functional Architecture
3-7
Hazard Analysis & Risk Assessment HARA
Formulation of Safety Goals
Safety Goals
Formulation of Timing Requirements
Timing Requirements
System Safety Design
Fault Tree Analysis (and others)
3-8
Inclusion of Safety Mechanisms
Review of Safety Requirements
Functional Architecture
Commercial Off-the-shelf Components
4-6
4-7
Decomposition
Component Selection
Components developed by Suppliers
Allocation to HW and SW
Segregation Requirements
Timing Requirements
Technical Safety Requirements
Further Safety Requirements
Safety Case
4-9
Construction of safety case
Assessment and Optimization
5-6
6-6
Requirement Refinement and Negotiation
Documentation and Reasoning on Design Decisions
System specific assessment and optimization
HW Selection and Parameterization
SW Refinement
6-7
System Model
Partitioning
Tasks
Mapping
Deployment
Code Generation
Execution Code
6-8
Static Analysis
Metrics
Other Simulation
Performance Simulation
Trace analysis
Execution + Tracing
Execution Trace
6-11
6-10
6-9
Legend:
Activity
Artifact

Figure 4.2: System Analysis (Phase 1)

- Papyrus (SysML): Requirements, Item Definition, Architecture
- ProR (ReqIF): Requirements, HARA
- Capella/System Modeling Workbench (Arcadia): Requirements, Architecture
- Siemens Capital Systems: Architecture

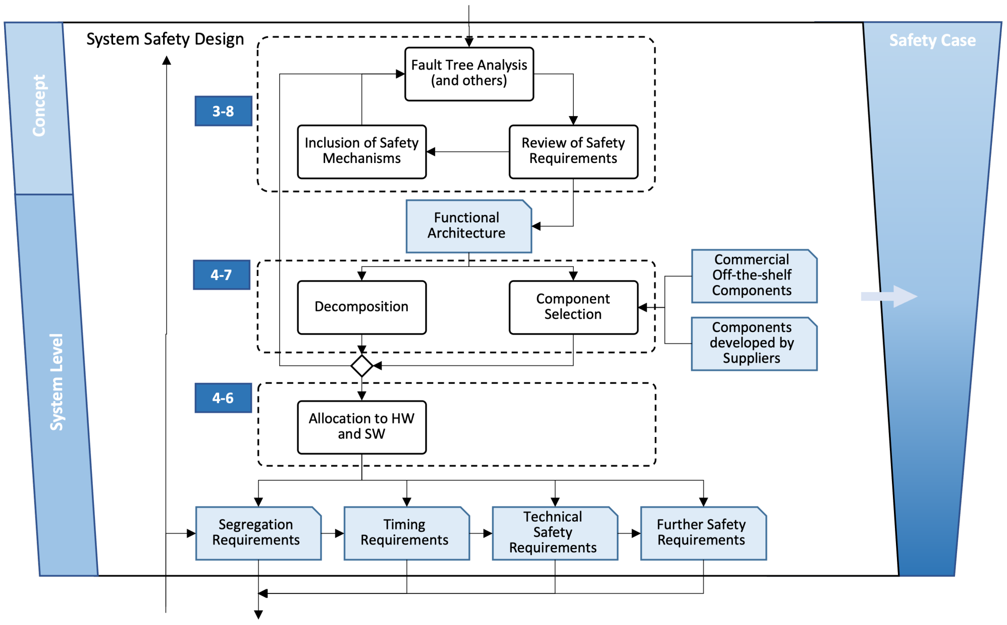

Concept
System Level
System Safety Design
Safety Case
3-8
4-7
4-6
Fault Tree Analysis (and others)
Inclusion of Safety Mechanisms
Review of Safety Requirements
Functional Architecture
Decomposition
Component Selection
Commercial Off-the-shelf Components
Components developed by Suppliers
Allocation to HW and SW
Segregation Requirements
Timing Requirements
Technical Safety Requirements
Further Safety Requirements

Figure 4.3: System Safety Design (Phase 2)

- Polarion: Requirements, Item Definition, HARA
- SafeTbox: Requirements, Architecture, HARA, Safety Case Modeling
- Vector PREEvision: Requirements, Item Definition, Architecture, HARA, Safety Case Modeling

4.2.2 System Safety Design

The second phase *System Safety Design* is shown in Figure 4.3. It is devoted to develop the (functional) system architecture and to derive the technical requirements for the following HW/SW design. As mentioned above, the construction of the functional safety concept is located in the concept phase of the ISO standard, whereas it belongs to the system design in the process presented here. The reason lies in the intended design flow of the design step: The respective safety requirements are derived in the *Fault Tree Analysis* and then reviewed (*Review Safety Requirements*) in order to derive suitable mitigation mechanisms. These mechanisms are directly integrated into the functional architecture model in the *Inclusion of Safety Mechanisms* activity, which thus can be seen as part of the system design. However, in ISO 26262, this step belongs to Clause 3-7 (Functional Safety Concept).

After this iterative design step is completed, the resulting architecture is decomposed and partitioned in order to prepare the subsequent preliminary allocation to hardware and software in the *Allocation to HW and SW* activity. The *Decomposition* activity is accompanied by a *Component Selection* activity, where components provided by suppliers and other off-the-shelf components are integrated into the design. The aim is to obtain an initial technical realization, often called “logical architecture”. These steps correspond to the System Design step (Clause 4-6) in the ISO standard. Additional virtual integration tests can be performed in the activity, which then corresponds to Clause 4-7.

In the *Allocation to HW and SW* activity, the selection of hardware elements of the system takes place as well as the allocation of the system components to software and hardware. This process imposes new requirements that need to be satisfied by the following phases in order to cover all top-level requirements. This includes safety related requirements such as segregation and diversity properties, as well as refined timing requirements, for example those related to maximal communication latencies. In AUTOSAR, this corresponds to the specification at the Virtual Function Bus (VFB) level [AUT17]. During this process, it may turn out that the initial architecture or previously derived requirements must be modified. Hence, earlier activities may be re-entered to incrementally refine the results if necessary.

Tools: The following list is purely informative, in no way complete, and not intended as recommendation. Each item contains the tool name, followed by the artifact format in parentheses, and the activities/work products.

- ProR (ReqIF): Functional Safety Concept
- APP4MC (AMALTHEA): Functional Safety Concept, Decomposition, Allocation
- Enterprise Architect (SysML): Functional Safety Concept, Decomposition, Allocation
- Papyrus (SysML): Functional Safety Concept, Decomposition, Allocation
- Capella/SMW (Arcadia): Functional Safety Concept, Decomposition, Allocation
- Polarion together with Siemens Capital Systems: Functional Safety Concept, Decomposition, Allocation
- SafeTbox: Functional Safety Concept, Fault Tree Analysis (and others), Decomposition, Allocation, Safety Case Modeling
- Vector PREEvision: Functional Safety Concept, Fault Tree Analysis (and others), Decomposition, Allocation, Technical Safety Concept, Safety Case Modeling
- Isograph FaultTree+: Fault Tree Analysis (and others)
- EMFTA: Fault Tree Analysis (and others)
- ALD Fault Tree Analyser: Fault Tree Analysis (and others)
- Fault Tree Analysis on R: Fault Tree Analysis (and others)

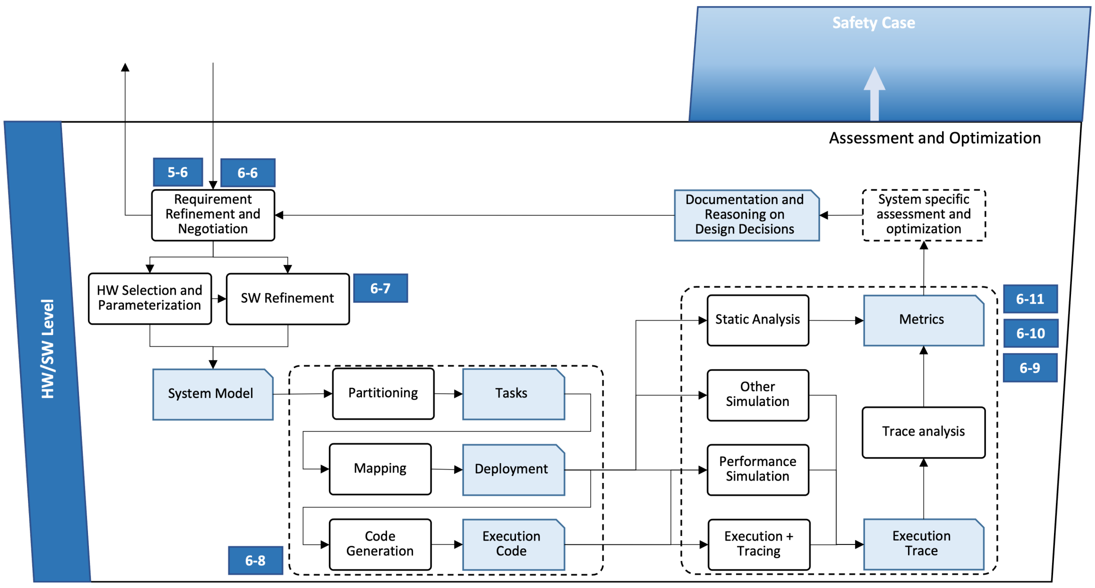


Figure 4.4: Assessment and Optimization (Phase 3)

4.2.3 Assessment and Optimization

The main outputs of the *System Analysis* and the *System Safety Design* phases, namely requirements and the technical safety architecture, serve as inputs of the final phase *Assessment and Optimization*. The corresponding design flow illustrated in Figure 4.4 is devoted to the systematic development, refinement, assessment and optimization of heterogeneous hardware/software designs.

Also this phase is designed as an iterative process. At the top left part of Figure 4.4 are the activities located that correspond to the specification of technical requirements and the software architecture design in the ISO standard (clauses 5-6, 6-6 and 6-7). Initially, the *Requirement Refinement and Negotiation* will typically pass the requirements from the previous design phase. Modifications to these requirements may take place in subsequent iterations. The assumption concerning the software architecture design is that the *HW Selection and Parameterization* may affect the developed software architecture, which calls for an optional *SW Refinement* activity.

The resulting architecture then is deployed, which includes the usual *Partitioning* and *Mapping* activities. The *Code Generation* activity aims at the implementation of the actual system functionality. This can happen manually or (semi-)automatically, e.g., using code generators. In the ISO standard, this corresponds to the Software Unit Design and Implementation (Clause 6-8).

The right part of Figure 4.4 shows the activities that are devoted to the assessment of the resulting technical architecture. The activities subsume all test, simulation and analysis methods that apply to this design level. The results of this evaluation phase provide the input to further system optimizations in the *System-specific Assessment and Optimization* activity, which may lead to further iteration loops depending on the overall assessment results.

The development of methods located at this design phase was a main focus of the PANORAMA project. For more information we refer to [Con20b; Con20a; Con20c]. Only shown abstractly in the figure is the fact that the evaluation results serve as a major input to the construction of the safety case (see, e.g., [KW04]).

Tools: The following list is purely informative, in no way complete, and not intended as recommendation. Each item contains the tool name, followed by the artifact format in parentheses, and the activities/work products.

- APP4MC (AMALTHEA): HW Selection and Parameterization, Partitioning, Mapping
- Artop (AUTOSAR): HW Selection and Parameterization, Partitioning, Mapping
- Siemens Capital VSTAR (AUTOSAR): HW Selection and Parameterization, Partitioning, Mapping

5 Engineering Scenarios

The engineering scenarios described within this chapter have been kindly provided by PANORAMA partners from different work packages, mainly WP1, WP3, and WP6. As described in Chapter 3, the partners were asked to fill in a survey template for engineering scenarios. In order to keep effort in check, some information had been gathered beforehand by the handbook authors from presentations, deliverables, and preliminary surveys within PANORAMA. The following engineering scenarios were collected and are described below:

- Change Impact Analysis (ES-1)
- Enhanced Project Development Life Cycle (ES-2)
- Verification of Consistency and Timing of End-to-End Chains (ES-3)
- Early Design Space Exploration (ES-4)
- Optimization of Task Deployment (ES-5)
- Verification of End-to-End Latency Requirements (ES-6)
- Analysis of Timing Behavior (ES-7)
- Correct Implementation of Safety Mechanisms (ES-8)
- Fault-Tolerant Scheduling Analysis (ES-9)
- Simulation of Timing Behavior (ES-10)
- Trace Analysis for Timing Behavior (ES-11)
- SystemC Performance Simulation (ES-12)

5.1 Change Impact Analysis (ES-1)

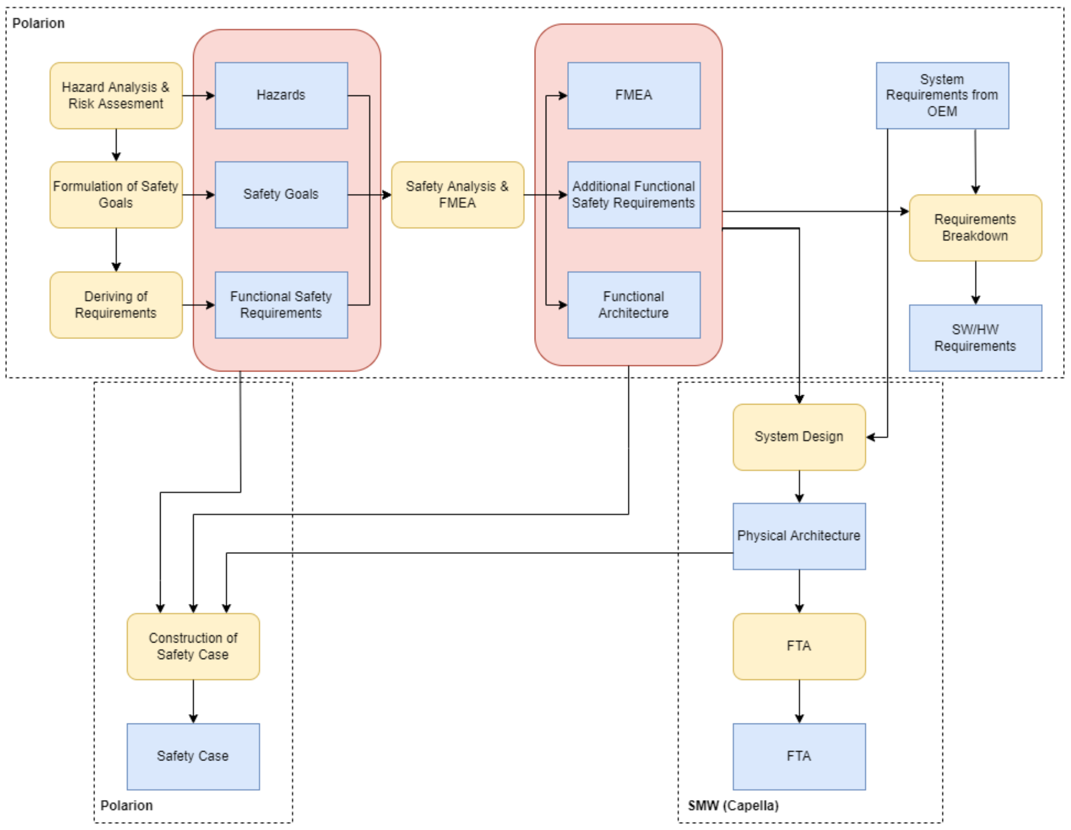
ES-1	Title: *Change Impact Analysis*
	Provider(s): *SIEMENS*

Goal

The engineering scenario can be described as a user story as follows:

> As a safety engineering / manager, I need to identify and asses the results of changes in terms of safety in order to ensure that the system remains safe. Usually, this is a manual task in which the all engineering artifacts need to be reviewed. Traceability enables an automated retrieval of artifacts which are affected by a specific change. This will ease the change impact analysis by reducing effort and the possibility of faults.

Activities in Focus



Input Artifacts	**Output Artifacts**
• *Hazards* identified during the Hazard & Risk Analysis • *Safety Goals* that are derived from the hazards • *Safety Requirements* derived from the safety goals which result from a safety analysis (e.g., an FMEA) • *System Requirements* defined by a customer • *Safety Analysis (FMEA & FTA)* conducted based on the functional or physical system architecture • *Functional/Physical Architecture* specified during the design process • *Safety Case* constructed based on the information from the design process	• *Hazards* changed or newly added (e.g., due to modified ASIL classification of a hazard) • *System Requirements* changed or newly added • *Safety Goals* which are affected by the change • *Safety Requirements* which are affected by the change • *Safety Analysis (FMEA & FTA)* which is affected by the change and must be adapted • *Functional/Physical Architecture* which is affected by the change • *Parts of the safety case* which are affected by the change
Traceability Information The relevant traceability information provided to the PANORover data set is displayed in Figure 5.1.	

Example Instantiation

1. Example Instantiation 1 (see Figure 5.2): A new customer requirement is introduced (e.g., enhancement of an ADAS functionality). This leads to new SW/HW requirements. A new HW component must be added in the system architecture, because the old one is not powerful enough to deal with the new task. As a result the safety analyses (FMEA & FTA) must be adapted to the new requirement and the modified system architecture. Moreover, the safety manager must check if the evidence in the safety case still fits the claims. If not, the safety case must be adapted accordingly (e.g., by adapting claims or adding new requirements).

2. Example Instantiation 2 (see Figure 5.3): The ASIL assessment of a hazard/safety goal must be adapted (e.g., after discussions with an assessor). As a result, a sensor A must be replaced with sensor B, which also involves the change of a supplier. As a result the safety analyses (FMEA & FTA) must be adapted to the modified safety requirements and the modified system architecture. Moreover, the respective claims in the safety case must be adapted and the safety manager must check if the evidence in the safety case still fits to the modified claims.

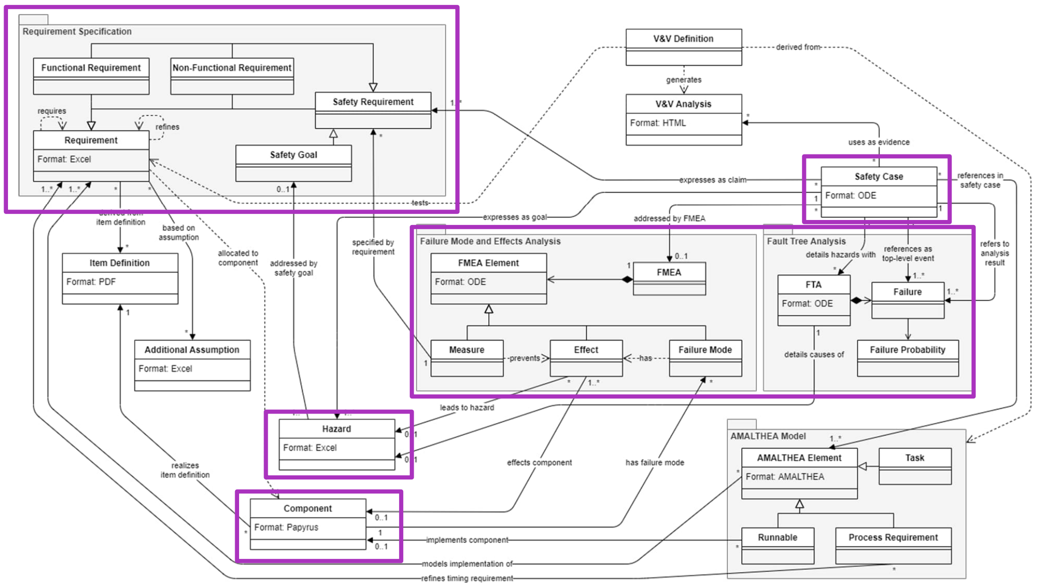


Figure 5.1: Traceability Information Used in ES-1

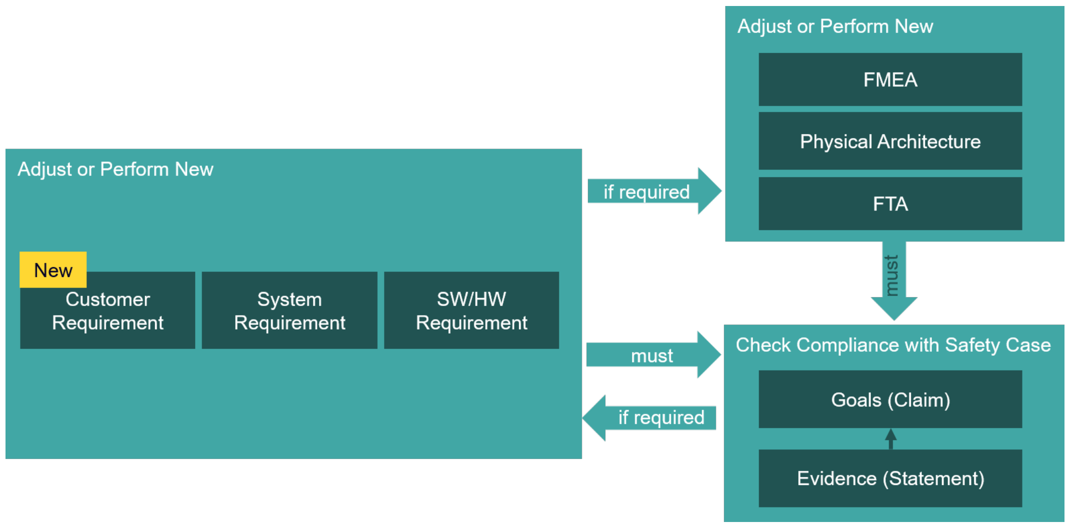


Figure 5.2: Example Instantiation 1 of ES-1

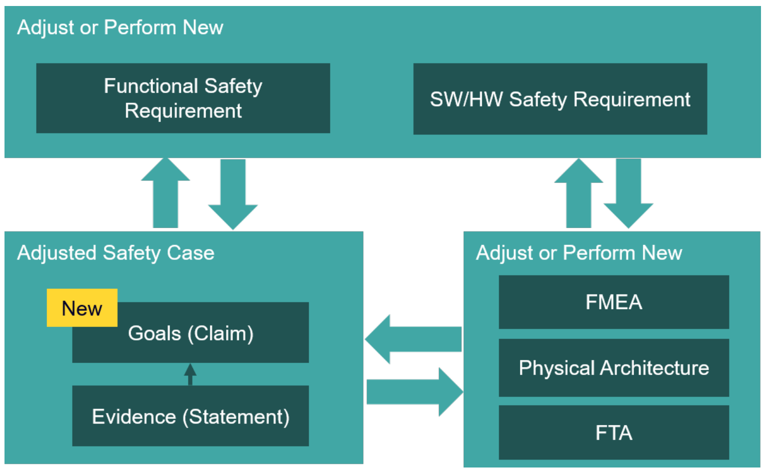


Figure 5.3: Example Instantiation 2 of ES-1

Table 5.1: Engineering Scenario ES-1

5.2 Enhanced Project Development Life Cycle (ES-2)

ES-2	Title: *Enhanced Project Development Life Cycle*	
	Provider(s): *Critical Software*	
Goal As a Software Architect/System Engineer/Functional Safety Engineer, I want to be able to evaluate how changes impact the system safety in early design phases. Typical document-based analysis requires a high manual effort and will not cope well with system changes. Integrating an MBSE methodology into our development process will allow iterating the system design and quickly evaluate the overall system safety.		
Activities in Focus Fault Tree Analysis (and others), Review of Safety Mechanisms, Inclusion of Safety Mechanisms, Static Analysis, System-specific Assessment and Optimization		

Input Artifacts
• AMALTHEA model annotated with safety-relevant information
  – Failure/Failure Mode
  – Failure Probability
	Output Artifacts
	• Failure propagation paths (between components)
	• FTA/FMEA (in ODE and Excel format)
	• Eclipse Capra with traceability links between AMALTHEA and ODE artifacts	

Traceability Information Eclipse Capra is used to map between AMALTHEA and ODE components in an attempt to simplify information sharing between partners.		

Example Instantiation
A possible instantiation of this engineering scenario can be seen in the following images, where it is assumed that the actor (Safety Engineer) uses the developed tool to load a previously populated AMALTHEA model, and then will perform and observe the following:

- The user navigates to the Architecture View where they will be presented with the system failure propagation paths, obtained from the relationship between all the defined AMALTHEA tasks/runnables (cf. Figure 5.4);
- The user annotates the AMALTHEA model with safety relevant information which was previously obtained from typical HARA activities (cf. Figure 5.5);
- The user selects the options to generate the safety related artifacts and is presented with dedicated FTA, FMEA and MCS views (cf. Figure 5.6);
- The user iterates the system design (adds a barrier) and checks in the FTA view that the probability of occurrence of the top/feared event decreased (cf. Figure 5.7).

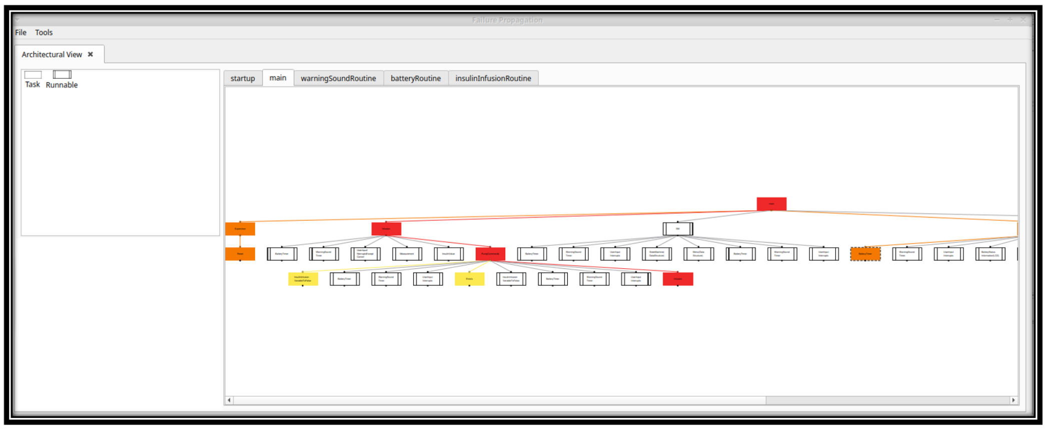
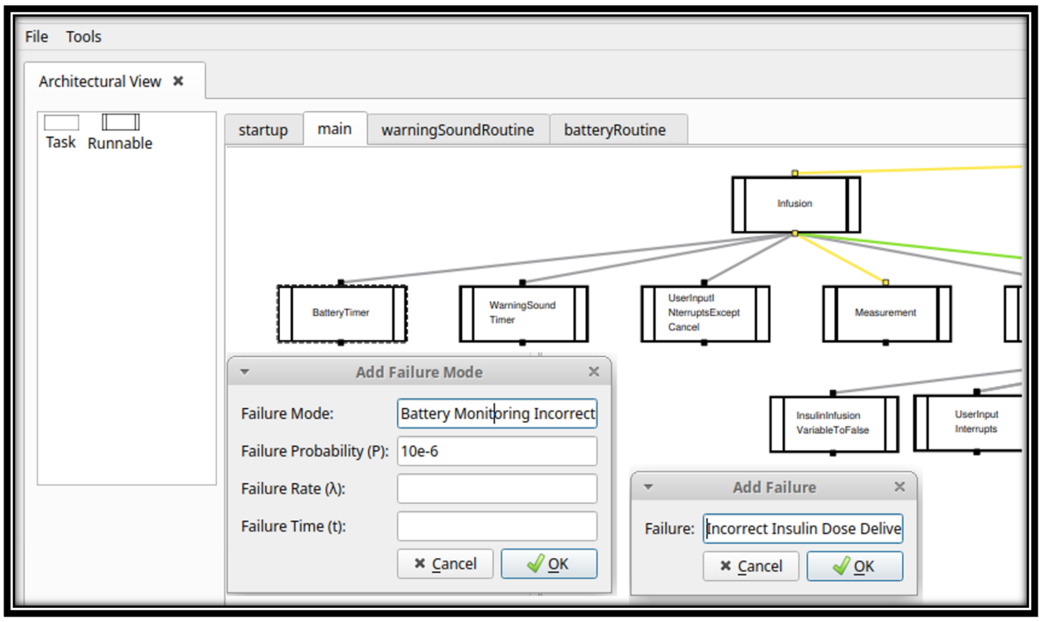


Figure 5.4: Example Instantiation of ES-2 (a) – Failure Propagation Paths (Architecture View)



Failure Mode: Battery Monitoring Incorrect
Failure Probability (P): 10e-6
Failure Rate (λ):
Failure Time (t):

Failure: Incorrect Insulin Dose Delive

Figure 5.5: Example Instantiation of ES-2 (b) – Annotation of AMALTHEA Models

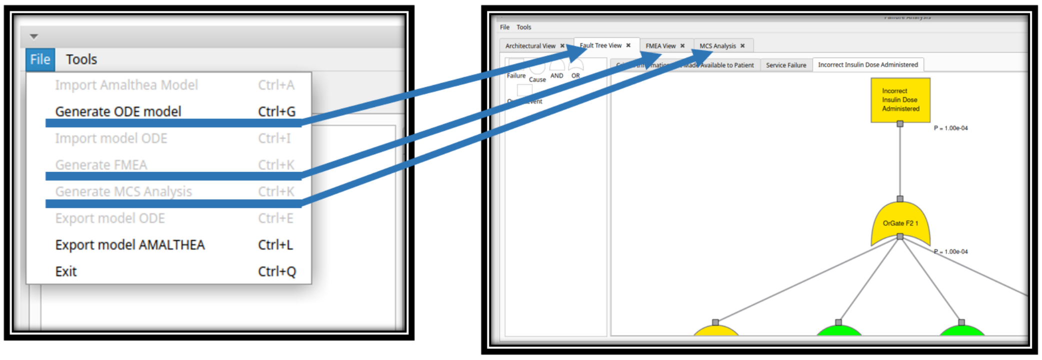


Figure 5.6: Example Instantiation of ES-2 (c) – FTA/FMEA/MCS Analysis Views

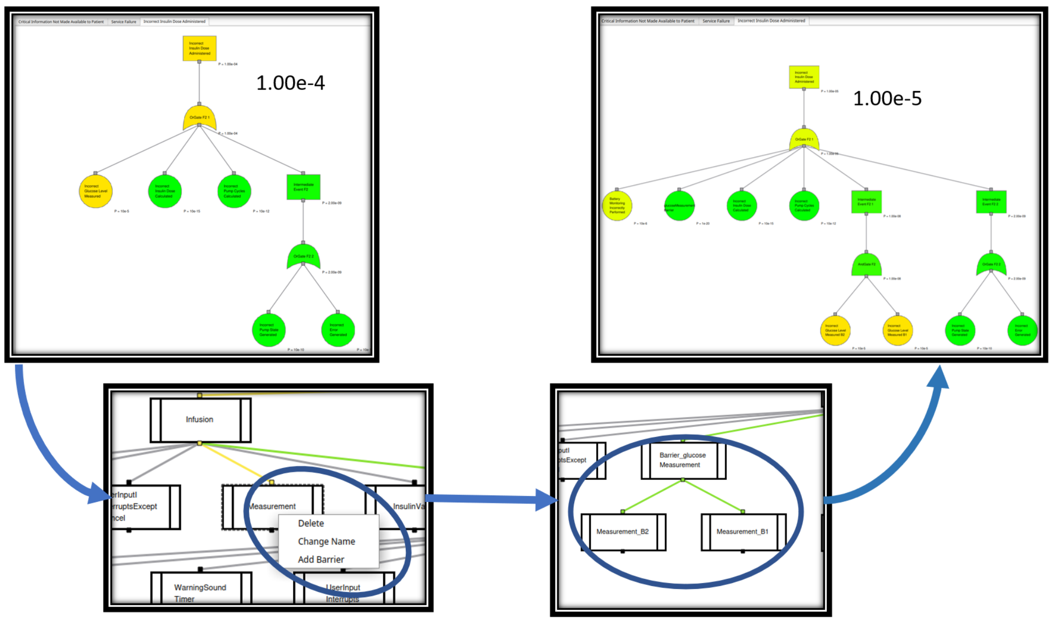


Figure 5.7: Example Instantiation of ES-2 (d) – Model Iteration and Result Visualization

Table 5.2: Engineering Scenario ES-2

5.3 Verification of Consistency and Timing of End-to-End Chains (ES-3)

ES-3	Title: *Verification of Consistency and Timing of End-to-End Chains*
	Provider(s): *INCHRON*

Goal

To ensure an accurate behavior in an automotive system (especially for autonomous driving), the complete data flow and processing of the information from the sensors up-to the actuators needs to be considered. This includes steps like preprocessing of sensor data, e.g., images from camera systems or radar, extraction of high-level information, e.g., other cars, pedestrians, or obstacles with their positions and movements, a combination of different information to a world picture, reasoning and decisions about actions based on this information and calculation of the resulting actions like trajectories to follow and steering commands to perform. Many of these steps need complex and time consuming calculations. The following aspects are of particular importance:

- the overall latency;
- the variation in the latency (jitter);
- the data ages, especially when merging information from different sources;
- the synchronicity of the different data sources; and
- the consistency of data processing, i.e., whether all data set are processed or the same set of data are processed multiple times.

Activities in Focus

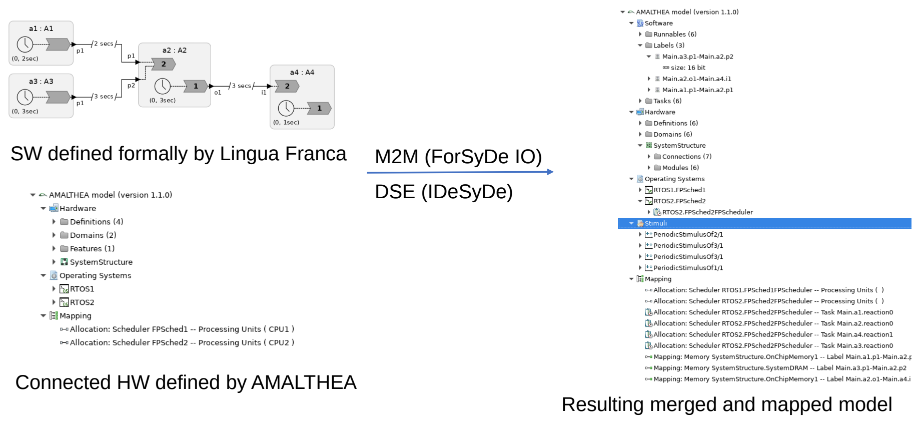
Hardware Selection and Parameterization, SW Refinement, Performance Simulation, Trace Analysis, Static Analysis

Input Artifacts	**Output Artifacts**
AMALTHEA model with the following information: • Software Model: – model of functions consisting of the: ∗ function nodes; ∗ the data dependencies between the function nodes; and ∗ forming event chains describing the data processing between input and output (without loops etc.). – the other (background) software possibly interfering with the function of above (e.g., located on the same resource); – the relevant timing information on the software mode, e.g., call frequency (period), jitter, activation relations etc. • Hardware model: – set of resources as ECUs, memory; – information on the scheduling; – mapping information.	• End-to-end response time • Data consistency satisfied • Information and visualization of system executions of interest (e.g., seen worst-case scenarios)
Traceability Information • Between (latency) requirements and the event chain; • Between event chain step and implementation of the (partly) function.	

Table 5.3: Engineering Scenario ES-3

5.4 Early Design Space Exploration (ES-4)

ES-4	Title: *Early Design Space Exploration*	
	Provider(s): *KTH*	
Goal The goal is to derive a mapping which satisfies the design's requirements. • Requirements include: timing, memory limits etc. • They must be proven during mapping via formal reasoning. • Mapping results can be further assessed with specific-purpose analysis tools.		
Activities in Focus Partitioning, Mapping		
Input Artifacts • AMALTHEA system model with "enough" information for analytical exploration. Enough information includes: – proper platform definition; – calculable WCETs; – proper scheduler definitions	**Output Artifacts** • The same AMALTHEA System model as before, but refined with new mapping entries and possible extra "tracing". • Estimates from the design space exploration can be optionally attached to the model.	
Example Instantiation The tool consists of model-to-model (M2M) transformations done with the ForSyDe IO tool and design space exploration (DSE) with the tool IDeSyDe. Figure 5.8 shows on the left hand side an example software model in the Lingua Franca format used by the ForSyDe IO and IDeSyDe tools internally, and an AMALTHEA hardware model. On the right hand side, the DSE result in form of a merged AMALTHEA model is shown that contains both the hardware and software model together with a mapping found by the DSE.		



AMALTHEA model (version 1.1.0)
- Hardware
  - Definitions (4)
  - Domains (2)
  - Features (1)
  - SystemStructure
- Operating Systems
  - RTOS1
  - RTOS2
- Mapping
  - Allocation: Scheduler FPSched1 -- Processing Units (CPU1)
  - Allocation: Scheduler FPSched2 -- Processing Units (CPU2)

AMALTHEA model (version 1.1.0)
- Software
  - Runnables (6)
  - Labels (3)
    - Main.a3.p1-Main.a2.p2
      - size: 16 bit
    - Main.a2.o1-Main.a4.i1
    - Main.a1.p1-Main.a2.p1
  - Tasks (6)
- Hardware
  - Definitions (6)
  - Domains (6)
  - SystemStructure
    - Connections (7)
    - Modules (6)
- Operating Systems
  - RTOS1.FPSched1
  - RTOS2.FPSched2
    - RTOS2.FPSched2FPScheduler
- Stimuli
  - PeriodicStimulusOf2/1
  - PeriodicStimulusOf3/1
  - PeriodicStimulusOf3/1
  - PeriodicStimulusOf1/1
- Mapping
  - Allocation: Scheduler RTOS1.FPSched1FPScheduler -- Processing Units ( )
  - Allocation: Scheduler RTOS2.FPSched2FPScheduler -- Processing Units ( )
  - Allocation: Scheduler RTOS2.FPSched2FPScheduler -- Task Main.a1.reaction0
  - Allocation: Scheduler RTOS2.FPSched2FPScheduler -- Task Main.a2.reaction0
  - Allocation: Scheduler RTOS2.FPSched2FPScheduler -- Task Main.a4.reaction1
  - Allocation: Scheduler RTOS2.FPSched2FPScheduler -- Task Main.a3.reaction0
  - Mapping: Memory SystemStructure.OnChipMemory1 -- Label Main.a1.p1-Main.a2.p
  - Mapping: Memory SystemStructure.SystemDRAM -- Label Main.a3.p1-Main.a2.p2
  - Mapping: Memory SystemStructure.OnChipMemory1 -- Label Main.a2.o1-Main.a4.i

Figure 5.8: Example Instantiation of ES-4

Table 5.4: Engineering Scenario ES-4

5.5 Optimization of Task Deployment (ES-5)

ES-5	Title: *Optimization of Task Deployment*
	Provider(s): *Dortmund University of Applied Sciences and Arts*
Goal Optimize a given task- to processor deployment such that the following is satisfied: • no task misses its deadline; and • event chain latencies are minimized.	
Activities in Focus Mapping, Deployment	
Input Artifacts AMALTHEA model with the following: • software model; • stimulus model; • hardware model; • OS model; and • initial mapping model (can be a random mapping but is needed to start the optimization process).	**Output Artifacts** AMALTHEA model containing the optimized deployment

Example Instantiation
An example instantiation of the engineering scenario is shown in Figure 5.9 and Figure 5.10. The figures show how the following steps that constitute the scenario are executed within the PANORAMA Cloud Service web frontend. This example shows how a model is optimized with the design space exploration (DSE) service such that end-to-end latencies of event chains in the provided input model are minimized.

1. Initial AMALTHEA model is provided in the UI manager.
2. The AMALTHEA model is then uploaded to the DSE optimization service which is configurable with different parameters like the optimization mode depending on the expected result.
3. Optimized AMALTHEA model is analyzed using the DSE analysis service to verify if it meets the requirements.
4. Metrics of the new model are compared with requirements. As can be seen in Figure 5.9, the deadline misses which are clearly large in the left chart are reduced to zero on the right chart. The left chart is a visualization of the timing metrics of a non-optimized AMALTHEA model and the right chart is that of the optimized AMALTHEA model. As is shown in Figure 5.10, the latencies are minimized in the bottom chart. The top chart represents the visualization of the end-to-end latency of a non-optimized AMALTHEA model, while the bottom chart shows that of the optimized model.

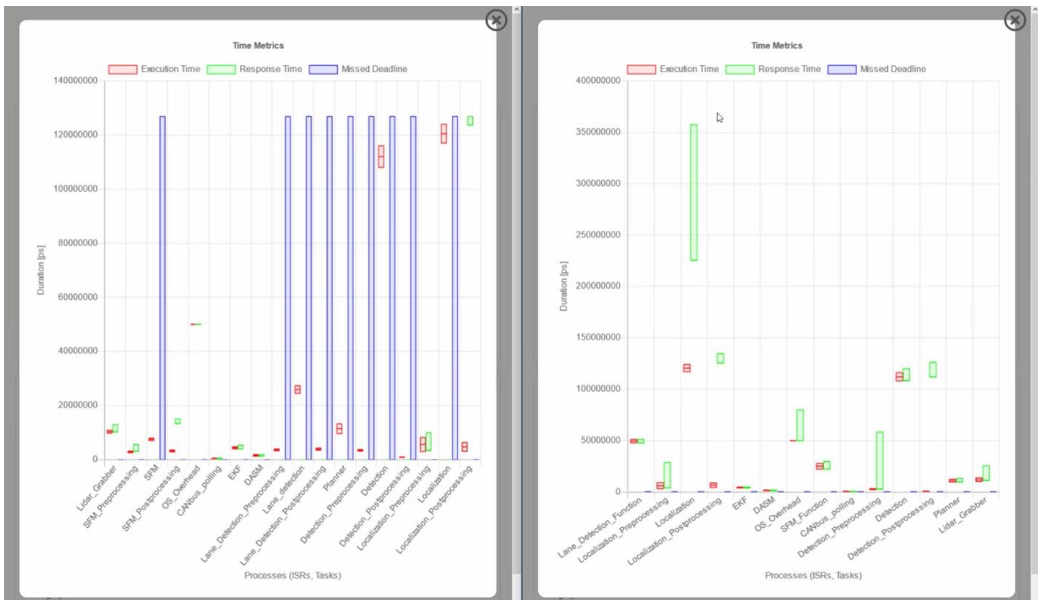


Figure 5.9: Example Instantiation of ES-5 (a)

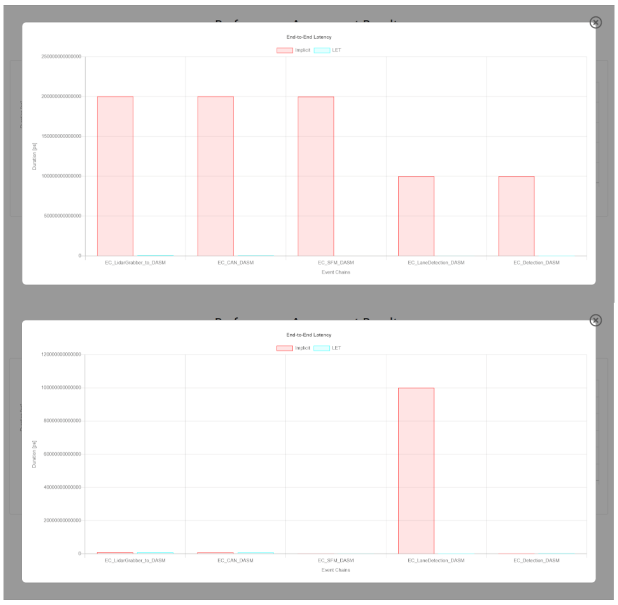


Figure 5.10: Example Instantiation of ES-5 (b)

Table 5.5: Engineering Scenario ES-5

5.6 Verification of End-to-End Latency Requirements (ES-6)

ES-6	Title: *Verification of End-to-End Latency Requirements*
	Provider(s): *Dortmund University of Applied Sciences and Arts*

Goal

Verify whether the system satisfies timing requirements (i.e., end-to-end latency of global functions), where:

- input is a mapping of software to hardware (tasks to cores, labels to memories); and
- output is an upper bound on event chain end-to-end latency.

End-to-end latency shall be assessed considering two different communication paradigms: implicit communication and Logical Execution Time (LET) communication.

Activities in Focus

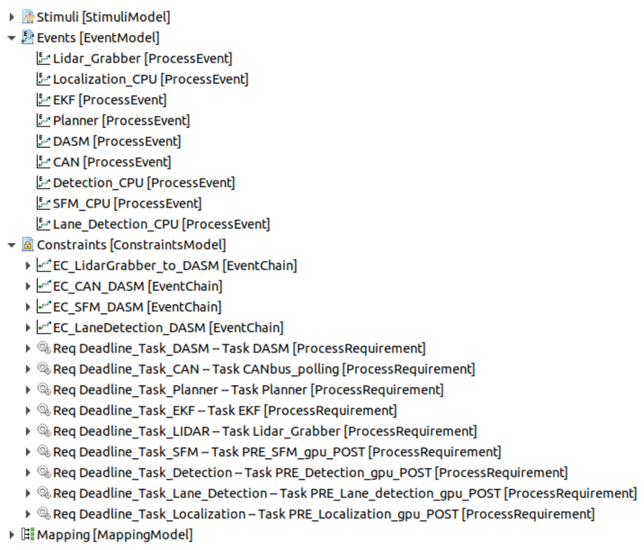
Static Analysis

Input Artifacts	**Output Artifacts**
AMALTHEA model with the following information: • SW model; • HW model; • stimulus model (to represent activation patterns of tasks); and • mapping model.	List of end-to-end latencies for all event chains, considering both implicit communication and LET communication

Example Instantiation

The engineering scenario shows the design space exploration (DSE) analysis of an AMALTHEA model, which is used to get information about the timing metrics and latency of the model. It consists of the following steps:

1. creating an AMALTHEA model (Figure 5.11);
2. identifying task communication (Figure 5.12);
3. identifying event chains (communication paths) (Figure 5.13); and
4. using the analysis cloud service (Figure 5.14)



```
▸ Stimuli [StimuliModel]
▾ Events [EventModel]
    Lidar_Grabber [ProcessEvent]
    Localization_CPU [ProcessEvent]
    EKF [ProcessEvent]
    Planner [ProcessEvent]
    DASM [ProcessEvent]
    CAN [ProcessEvent]
    Detection_CPU [ProcessEvent]
    SFM_CPU [ProcessEvent]
    Lane_Detection_CPU [ProcessEvent]
▾ Constraints [ConstraintsModel]
  ▸ EC_LidarGrabber_to_DASM [EventChain]
  ▸ EC_CAN_DASM [EventChain]
  ▸ EC_SFM_DASM [EventChain]
  ▸ EC_LaneDetection_DASM [EventChain]
  ▸ Req Deadline_Task_DASM – Task DASM [ProcessRequirement]
  ▸ Req Deadline_Task_CAN – Task CANbus_polling [ProcessRequirement]
  ▸ Req Deadline_Task_Planner – Task Planner [ProcessRequirement]
  ▸ Req Deadline_Task_EKF – Task EKF [ProcessRequirement]
  ▸ Req Deadline_Task_LIDAR – Task Lidar_Grabber [ProcessRequirement]
  ▸ Req Deadline_Task_SFM – Task PRE_SFM_gpu_POST [ProcessRequirement]
  ▸ Req Deadline_Task_Detection – Task PRE_Detection_gpu_POST [ProcessRequirement]
  ▸ Req Deadline_Task_Lane_Detection – Task PRE_Lane_detection_gpu_POST [ProcessRequirement]
  ▸ Req Deadline_Task_Localization – Task PRE_Localization_gpu_POST [ProcessRequirement]
▸ Mapping [MappingModel]
```

Figure 5.11: Example Instantiation of ES-6 (a)

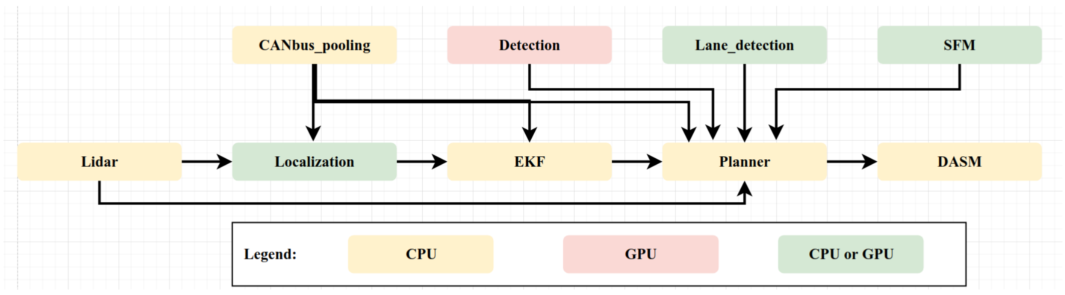


Figure 5.12: Example Instantiation of ES-6 (b)

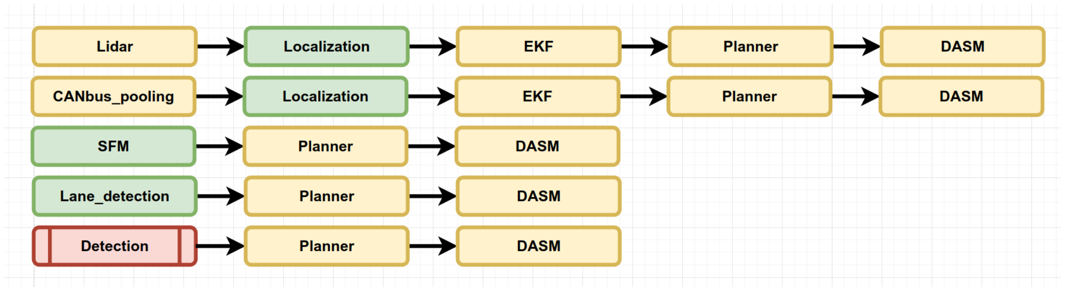


Figure 5.13: Example Instantiation of ES-6 (c)

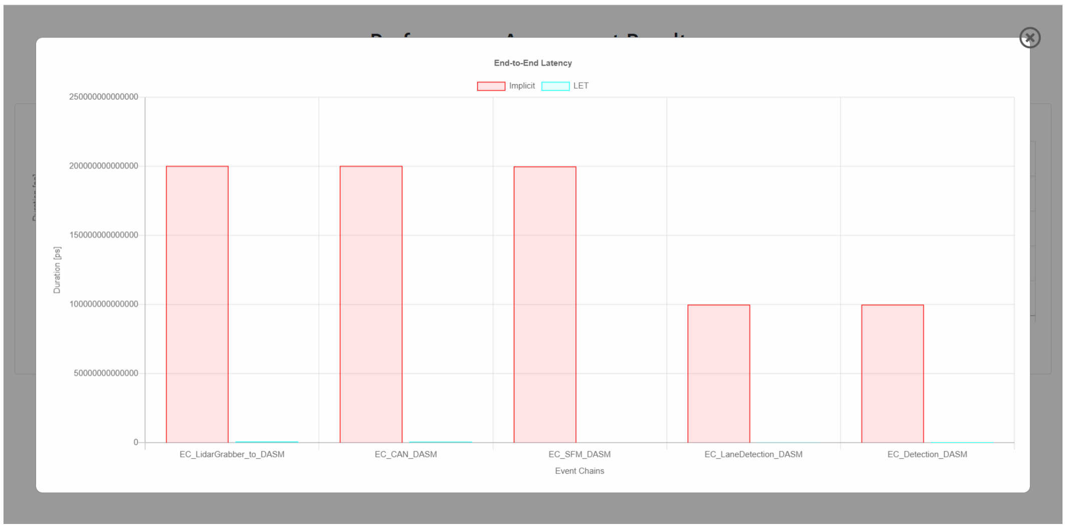


Figure 5.14: Example Instantiation of ES-6 (d)

Table 5.6: Engineering Scenario ES-6

5.7 Analysis of Timing Behavior (ES-7)

ES-7	Title: *Analysis of Timing Behavior*
	Provider(s): *INCHRON*

Goal

- The goal is to calculate and guarantee the timing behavior especially early in the design process.
- Identify critical situations and calculate (reasonable) upper bounds which are suitable for deployment.
  - Get guarantees for the worst-case behavior for all possible scheduling situations.
  - Check the scheduling and interaction of components early without having the complete implementation ready for deployment.
  - Identify spare capacity and test the possibilities for integration of additional functionality.
  - Identify (and optimize) the most critical scheduling constellations.
- Guarantee the correct behavior for example for aspects like:
  - the overall latency;
  - the variation in the latency (jitter);
  - the data ages, especially when merging information from different sources;
  - the synchronicity of the different data sources; and
  - the consistency of data processing, so whether all data sets are processed or if the same set of data is processed multiple times.

Activities in Focus
Static Analysis

Input Artifacts	**Output Artifacts**
AMALTHEA model with the following information: • Software Model: – Model of functions consisting of the: ∗ function nodes; ∗ the data dependencies between the function nodes; and ∗ forming event chains describing the data processing between input and output (without loops etc.) – The other (background) software possibly interfering with the function of above (e.g located on the same Resource). – The relevant timing information on the software mode e.g. call frequency (period), jitter, activation relations etc. • Hardware model: – set of resources as ECUs, memory; – information on the scheduling; and – mapping information	• End-to-end response time • Data consistency satisfied • Information and Visualization of system executions of interest (e.g., seen worst-case scenarios)
Traceability Information • Between (latency) requirements and the event chain. • Between event chain step and implementation of the (partly) function.	

Table 5.7: Engineering Scenario ES-7

5.8 Correct Implementation of Safety Mechanisms (ES-8)

ES-8	Title: *Correct Implementation of Safety Mechanisms*
	Provider(s): *OFFIS*

Goal

The goal is to construct evidence that safety mechanisms are correctly realized in the technical architecture. Initially, safety requirements and solutions are captured in the safety case. Potential failures that endanger safety requirements are modeled by a fault tree. Safety mechanisms are implemented along the system architecture and the technical system model. In the technical architecture, failure detection and mitigation are refined with respect to timing, which introduces new safety requirements. Timing analysis verifies that these requirements are effectively implemented (wrt. timing and mode changes). Therefore, the analysis provides evidence for correct implementation of the selected safety mechanisms.
The analysis is detailed below.

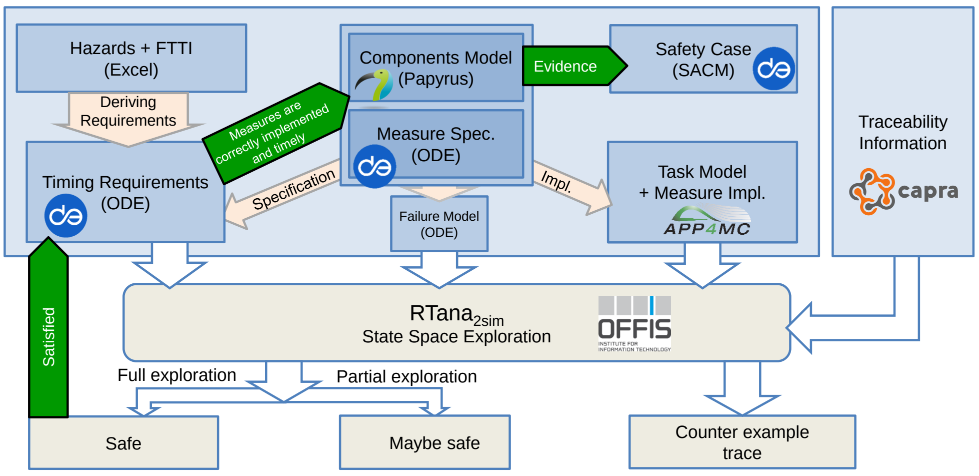
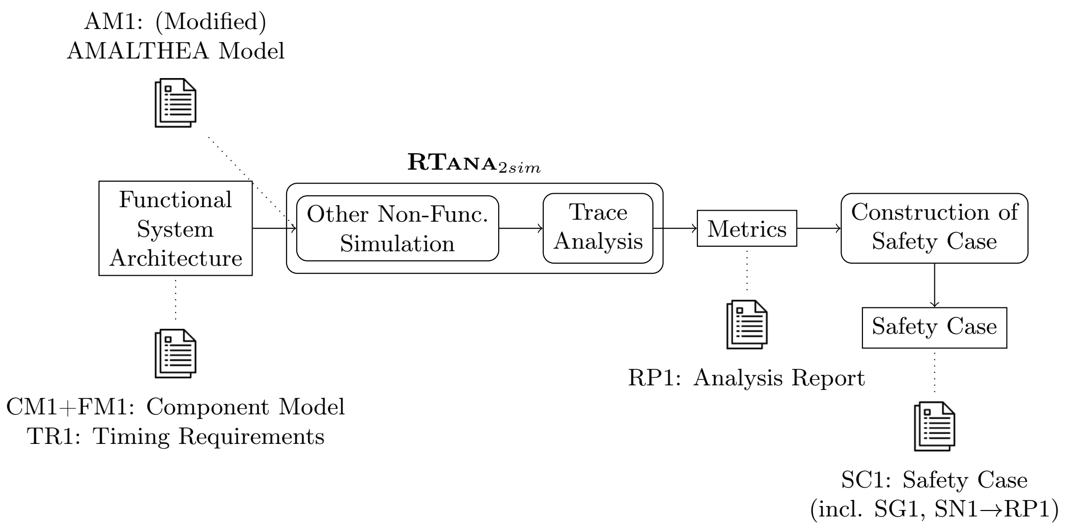


Figure 5.15: Structure of ES-8

Activities in Focus

The engineering scenario focuses on the *Other Non-Functional Simulation* and *Trace Analysis* activities from the *Assessment and Optimization* phase.


AM1: (Modified) AMALTHEA Model
RTANA$_{2sim}$
Functional System Architecture
Other Non-Func. Simulation
Trace Analysis
Metrics
Construction of Safety Case
Safety Case
CM1+FM1: Component Model
TR1: Timing Requirements
RP1: Analysis Report
SC1: Safety Case (incl. SG1, SN1→RP1)

Input Artifacts

- *Component Model* (CM1) that models the (updated) system architecture after the safety process has been completed; carrier of assigned *Timing Requirements* (TR1) to establish traceability.
- *AMALTHEA Model* (AM1) that complies with the system safety concept modeled in CM1.
- *Safety Requirements* (SR1) that specify a selected safety mechanism as a *Solution* (SN1) to mitigate a previously identified risk.
- *Failure Model* (FM1) that results from a fault tree analysis and is based on the updated *Component Model* (CM1) with failures modes.

Output Artifacts

- *Analysis Report* (RP1) with results related to the *Timing Requirements* (TR1) that is integrated in the *Safety Case* (SG1).

Traceability Information

The relevant traceability information is shown below.

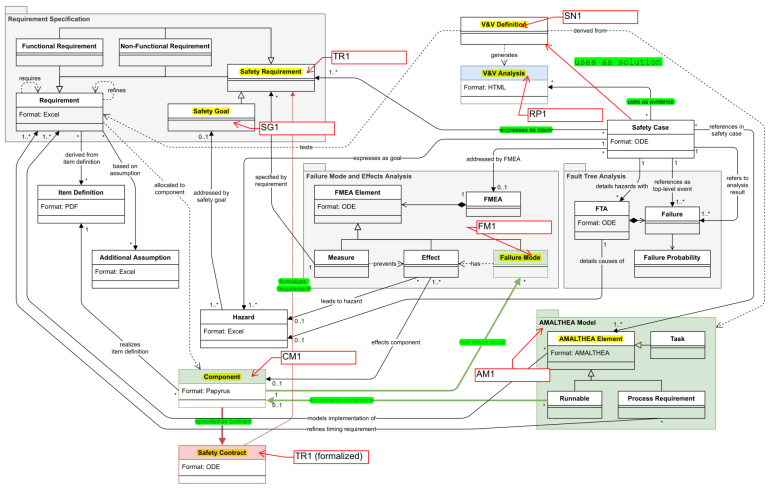
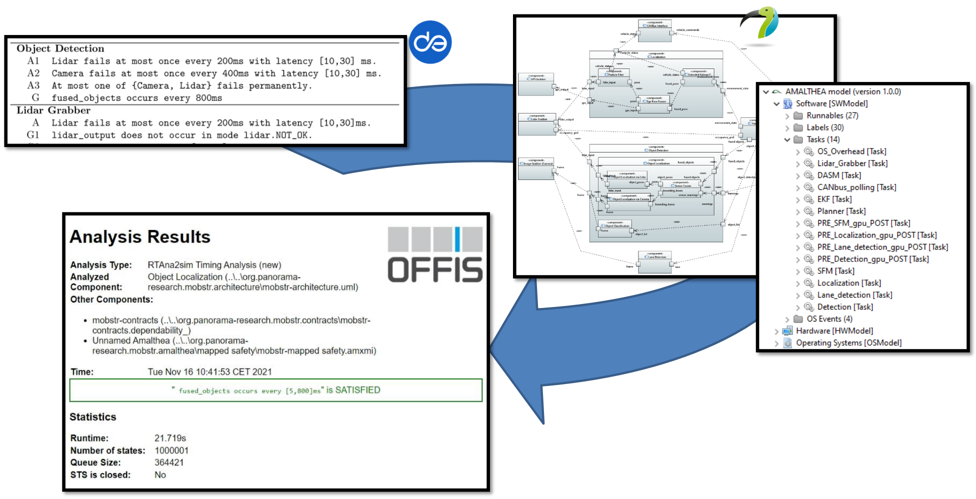


Figure 5.16: Relevant traceability information for ES-8

Example Instantiation

An example instantiation of the engineering scenario is illustrated below. The figure shows different artifacts from the MobSTr dataset that are visible in the scenario: timing requirements, the component model, the AMALTHEA model, as well as an analysis report.



Object Detection	
A1	Lidar fails at most once every 200ms with latency [10,30] ms.
A2	Camera fails at most once every 400ms with latency [10,30] ms.
A3	At most one of {Camera, Lidar} fails permanently.
G	fused_objects occurs every 800ms
Lidar Grabber	
A	Lidar fails at most once every 200ms with latency [10,30]ms.
G1	lidar_output does not occur in mode lidar.NOT_OK.

- AMALTHEA model (version 1.0.0)
  - Software [SWModel]
    - Runnables (27)
    - Labels (30)
    - Tasks (14)
      - OS_Overhead [Task]
      - Lidar_Grabber [Task]
      - DASM [Task]
      - CANbus_polling [Task]
      - EKF [Task]
      - Planner [Task]
      - PRE_SFM_gpu_POST [Task]
      - PRE_Localization_gpu_POST [Task]
      - PRE_Lane_detection_gpu_POST [Task]
      - PRE_Detection_gpu_POST [Task]
      - SFM [Task]
      - Localization [Task]
      - Lane_detection [Task]
      - Detection [Task]
    - OS Events (4)
  - Hardware [HWModel]
  - Operating Systems [OSModel]

Analysis Results

Analysis Type:	RTAna2sim Timing Analysis (new)
Analyzed Component:	Object Localization (..\..\org.panorama-research.mobstr.architecture\mobstr-architecture.uml)
Other Components:	

- mobstr-contracts (..\..\org.panorama-research.mobstr.contracts\mobstr-contracts.dependability_)
- Unnamed Amalthea (..\..\org.panorama-research.mobstr.amalthea\mapped safety\mobstr-mapped safety.amxmi)

Time: Tue Nov 16 10:41:53 CET 2021

" fused_objects occurs every [5,800]ms" is SATISFIED

Statistics

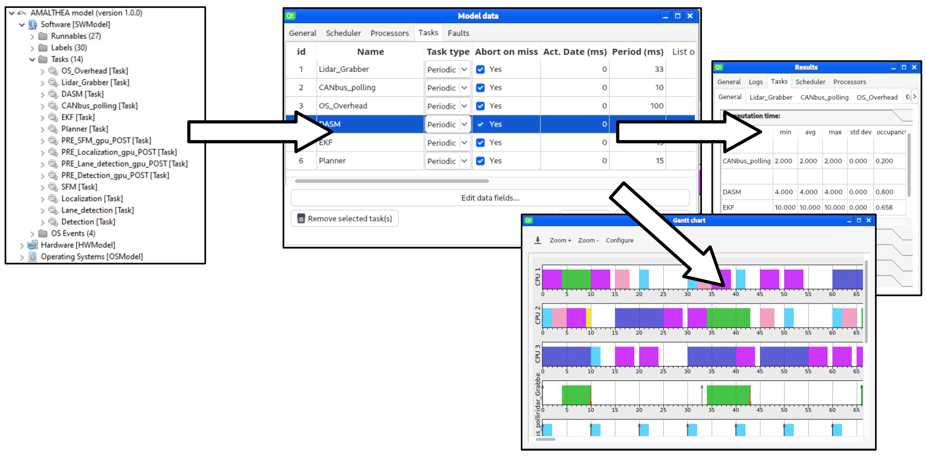
Runtime:	21.719s
Number of states:	1000001
Queue Size:	364421
STS is closed:	No

Figure 5.17: Example instantiation of ES-8

Table 5.8: Engineering Scenario ES-8

5.9 Fault-Tolerant Scheduling Analysis (ES-9)

ES-9	Title: *Fault-Tolerant Scheduling Analysis*
	Provider(s): *OTH Regensburg*
Goal The goal is to provide the possibility of injecting faults and randomness of task parameters into the simulation process of real-time scheduling algorithms. By doing so, deeper insight about the robustness of the system under test can be gained and the resilience of scheduling algorithms against faults can be evaluated. The measured response time distributions, number of missed deadlines as well as preemptions and migrations provide levels of confidence for the safety of the system.	
Activities in Focus The engineering scenario focuses on the *Performance Simulation* activities from the *Assessment and Optimization* phase.	
Input Artifacts • *Component Model* that models the system architecture, e.g., from the *AMALTHEA Model*. • *Failure Model* that models the possible faults and their possibilities based on the *Component Model*.	**Output Artifacts** • *Analysis Report* with results related to the simulation process, e.g., performance metrics and execution trace.
Traceability Information The optional *Failure Model* input can be obtained from a fault tree analysis of the specified *Component Model*. The aforementioned input artifacts are consumed to provide an *Analysis Report* of the simulated environment.	
Example Instantiation An example instantiation of the engineering scenario is illustrated in Figure 5.18. The figure shows an example input *AMALTHEA model* on the left, and different parts of the FiSimSo user interface: An editor for configuring model data, a Gantt chart viewer for simulation traces and an editor for inspecting and evaluating analysis results. In practice, the user specifies the *AMALTHEA* input file which is parsed by FiSimSo in order to perform the simulation. After the simulation has finished, FiSimSo gives insight about various performance metrics and the execution trace of the simulated system.	



AMALTHEA model (version 1.0.0)
- Software [SWModel]
  - Runnables (27)
  - Labels (30)
  - Tasks (14)
    - OS_Overhead [Task]
    - Lidar_Grabber [Task]
    - DASM [Task]
    - CANbus_polling [Task]
    - EKF [Task]
    - Planner [Task]
    - PRE_SFM_gpu_POST [Task]
    - PRE_Localization_gpu_POST [Task]
    - PRE_Lane_detection_gpu_POST [Task]
    - PRE_Detection_gpu_POST [Task]
    - SFM [Task]
    - Localization [Task]
    - Lane_detection [Task]
    - Detection [Task]
  - OS Events (4)
- Hardware [HWModel]
- Operating Systems [OSModel]

Model data

General Scheduler Processors Tasks Faults

id	Name	Task type	Abort on miss	Act. Date (ms)	Period (ms)	List o
1	Lidar_Grabber	Periodic	✓ Yes	0	33	
2	CANbus_polling	Periodic	✓ Yes	0	10	
3	OS_Overhead	Periodic	✓ Yes	0	100	
[illegible]	DASM	Periodic	✓ Yes	0	[illegible]	
[illegible]	EKF	Periodic	✓ Yes	0	[illegible]	
6	Planner	Periodic	✓ Yes	0	15	

Edit data fields...

Remove selected task(s)

Results

General Logs Tasks Scheduler Processors

General Lidar_Grabber CANbus_polling OS_Overhead

Computation time:

	min	avg	max	std dev	occupancy
CANbus_polling	2.000	2.000	2.000	0.000	0.200
DASM	4.000	4.000	4.000	0.000	0.800
EKF	10.000	10.000	10.000	0.000	0.658

Figure 5.18: Example Instantiation of ES-9

Table 5.9: Engineering Scenario ES-9

5.10 Simulation of Timing Behavior (ES-10)

ES-10	Title: *Simulation of Timing Behavior*
	Provider(s): *INCHRON*

Goal

- Test and understand the timing behavior early in the design process and through the complete design process and the life-cycle of the systems.
- Identify the root-cause for observed situations and critical timing problems.
- Specific goals:
  - identify critical situations early in the development process;
  - test the scheduling and interaction of components early without having the complete implementation ready for deployment; and
  - identify spare capacity and test the possibilities for integration of additional functionality.
- Guarantee the correct behavior for example for aspects like:
  - the overall latency;
  - the variation in the latency (jitter);
  - the data ages, especially when merging information from different sources;
  - the synchronicity of the different data sources; and
  - the consistency of data processing, i.e., whether all data sets are processed or the same set of data is processed multiple times.

Activities in Focus
Performance Simulation, Trace Analysis

Input Artifacts	**Output Artifacts**
AMALTHEA model with the following information: • Software Model: – Model of functions consisting of the: ∗ function nodes; ∗ the data dependencies between the function nodes; ∗ forming event chains describing the data processing between input and output (without loops etc.). – The other (background) software possibly interfering with the function of above (e.g., located on the same resource). – The relevant timing information on the software mode, e.g., call frequency (period), jitter, activation relations etc. • Hardware model: – set of resources as ECUs, memory; – information on the scheduling; and – mapping information	• End-to-end response time. • Data consistency satisfied. • Information and visualization of system executions of interest (e.g. seen worst-case scenarios).
Traceability Information • Between (latency) requirements and the event chain. • Between event chain step and implementation of the (partly) function.	

Table 5.10: Engineering Scenario ES-10

5.11 Trace Analysis for Timing Behavior (ES-11)

ES-11	Title: *Trace Analysis for Timing Behavior*
	Provider(s): *INCHRON*

Goal

- The goal is to test, visualize and understand the timing behavior of a system with an executable.
- Specific goals:
  - understand the root-cause for observed timing behavior;
  - refine timing model with measured information.
- Aspects to visualize are:
  - task, process execution, states and events;
  - bus messages and communication flows;
  - event chains; and
  - system load for various resources

Activities in Focus
Execution on Hardware + Tracing, Trace Analysis

Input Artifacts	**Output Artifacts**
• Measured timing information: – from a single resource; – from multiple interconnected resource running concurrently (leading to multiple synchronized traces). • Model information: – if available; – name of processes tasks etc.; – scheduling information; – *Helps in the interpretation of the trace.* • Requirements	• Trace visualization. • Statistic information on trace. • Requirement evaluation.

Traceability Information
• Between (latency) requirements and the event chain. • Between event chain step and implementation of the (partly) function.

Table 5.11: Engineering Scenario ES-11

5.12 SystemC Performance Simulation (ES-12)

ES-12	Title: *SystemC Performance Simulation*
	Provider(s): *University of Rostock, Robert Bosch GmbH*

Goal

This engineering scenario addresses two phases of the development life cycle.

- Definition Phase:
  - Prerequisites:
    - Full software is not available, but functionality is known and runtimes can be roughly estimated.
    - OR: software integration is pending but functions are implemented, so they can be measured in isolation on physical hardware or virtual prototype.
  - Task: Early exploration of variants of the integrated system
    - What is the preemption pattern of tasks?
    - Are there extensive waiting times, due to shared resources?
    - What happens if the cache hit rate drops drastically?
    - What is the impact if hardware features change? (In case hardware system is not fixed)
- During integration:
  - Prerequisites:
    - Full software is available and can be run on target, runtimes (net-execution time) are available.
    - Integrated system does not behave as expected, e.g., due to resource conflict.
  - Task: Use simulation to find promising candidates to test on target (exploration space reduction)

Inputs to evaluate:

- OS setup:
  - scheduling algorithm;
  - core affinities;
  - priorities.
- Hardware features (definition phase only):
  - memory bandwidth and access latencies;
  - effect of using suitable cores for a function (e.g., CPU vs. DSP).
- Data locality in memory/memories

Solution:

- Performance Simulation (Tool: APP4MC.sim):
  - adaptable to specific needs;
  - freely available (Open Source);
  - based on SystemC.
- Timing-Accurate Simulation of virtual ECUs:
  - on non-realtime systems (PCs);
  - timing-accurate functional simulation;
  - coupling of functional and timing simulation via FMI 3.

Involved tools:

- Eclipse APP4MC to define system model in AMALTHEA.
- APP4MC.sim to perform non-functional timing simulation.
- Postprocessing/evaluation of timing traces (not in scope of Eclipse APP4MC or APP4MC.sim).

Activities in Focus
Code Generation, Performance Simulation

Input Artifacts	**Output Artifacts**
• AMALTHEA System Model	• Execution Trace

Example Instantiation

The engineering scenario instantiation consists of two parts:

1. *Performance Simulation.* Here, execution traces are generated out of an AMALTHEA model with the help of a SystemC simulation (see Figure 5.19).
   a) A model2text transformation generates executable SystemC code out of an AMXMI file that models the timing behavior.
   b) A CMake build configuration drives the compilation and linking of the executable binary.
   c) The binary is executed and outputs one or more execution traces that can be further processed or visualized.

2. *Timing Accurate Simulation of Virtual ICUs.* This part consists of three components (Figure 5.20, left):
   - A *Timing Simulation* is responsible for the correct scheduling and accurate timing. It generates timing and scheduling events for the simulation.
   - The *Functional Simulation* is responsible for functional accuracy and generates communication events.
   - The *Communication Point Service* interfaces between those two components and manages data consistency in the simulation.

   This architecture can be used either as an open-loop simulation where the timing simulation can be driven by a pre-computed or recorded execution trace, or as a closed-loop simulation (Figure 5.20, right) where the output of the functional simulation (e.g mode changes) is fed back to the timing simulation component.

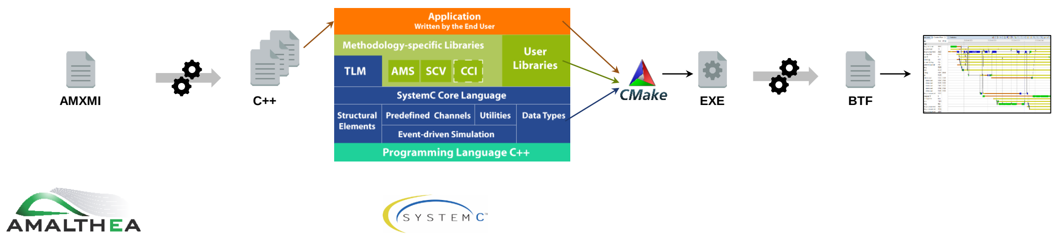


Figure 5.19: Example Instantiation of ES-12 (a) – Performance Simulation

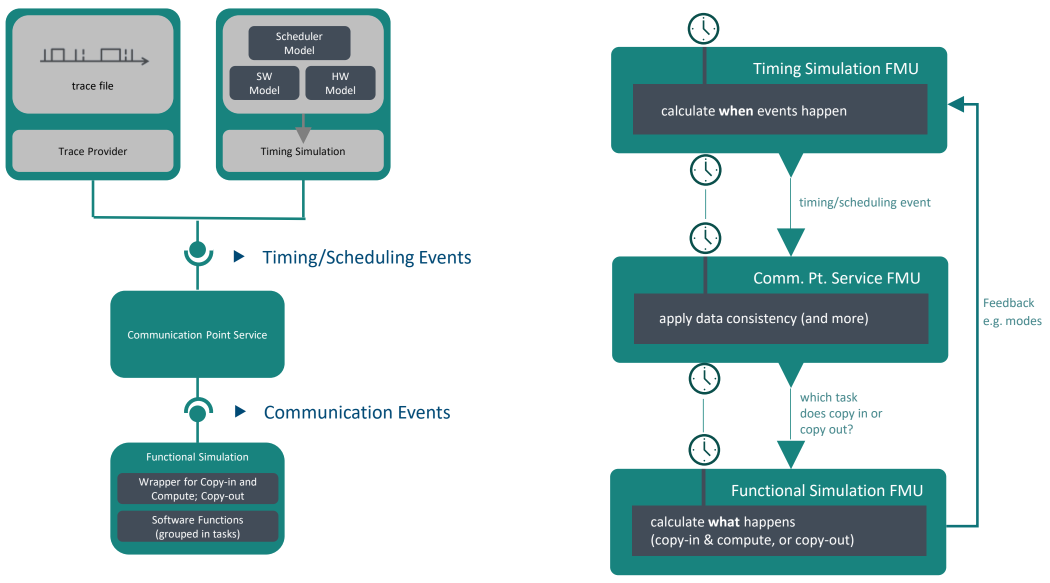

trace file
Trace Provider
Scheduler Model
SW Model
HW Model
Timing Simulation
Timing/Scheduling Events
Communication Point Service
Communication Events
Functional Simulation
Wrapper for Copy-in and Compute; Copy-out
Software Functions (grouped in tasks)
Timing Simulation FMU
calculate **when** events happen
timing/scheduling event
Comm. Pt. Service FMU
apply data consistency (and more)
which task does copy in or copy out?
Functional Simulation FMU
calculate **what** happens (copy-in & compute, or copy-out)
Feedback e.g. modes

Figure 5.20: Example Instantiation of ES-12 (b) – Timing-Accurate Simulation of Virtual ECUs: Architecture (left) and Closed-loop Simulation (right)

Table 5.12: Engineering Scenario ES-12

5.13 Other Tools

Two additional tools that have not been mentioned explicitly in the engineering scenarios are considered in this work.

ComposR This tool provided by SIEMENS has been used within the MobSTr demonstrator to perform the FTA and FMEA. The analysis results can be exported in ODE format.

Syntactic Load Generator This prototype tool provided by BOSCH uses the model-to-text transformation framework from Eclipse APP4MC to automatically generate executable code from AMALTHEA models. This code does not provide functionality, but contains generic instructions that cause a CPU load that matches the execution times specified in AMALTHEA. This way, the real-time behavior can be evaluated on real hardware. Alternatively, real-time critical software can be tested together with third-party software on a system without leaking intellectual property of that third party, as it would be the case when using the real software. In the MobSTr dataset, the Syntactic Load Generator has been used to provide example code for the system under development.

6 Discussion

Chapter 4 is devoted to research question **RQ-1** “What must a system engineering process look like to enable the standard-compliant development of heterogeneous systems?”. Although our suggestion must not be considered as *the one and only possible* answer, it is based on a careful analysis of previous work (see, e.g., [TMSP16]), and involves the expertise from academic and industrial partners in two successive projects. It is also not complete, as it focusses mainly on the engineering activities. Most of the management activities such as planning, status surveillance and reporting are not considered. On the other hand, it is quite complete with respect to the relation with the core engineering activities of the ISO 26262 standard, with the exception of the product-level validation activities (Clauses 4-7 and 4-8).

Chapter 5 is concerned with the first part of **RQ-2** “What are typical engineering scenarios?”. Indeed, the selection is also in no way complete and reflects the development in the context of the PANORAMA project. However, all engineering scenarios have been implemented, enabling us to put these findings together in order to give a (partial) answer to **RQ-3** “Which tools can be used to implement the process?” and “How can tool interoperability be ensured along the process?”. The second part of **RQ-2** “How does the process support these scenarios?” is self-evident: Defining and structuring the required design activities, and thus identifying their interfaces provides guidance on how these scenarios shall be realized. We will come back to **RQ-4** “How can applicability of the process be ensured?” in Section 6.3.

For the synthesis of results, the engineering scenarios, as well as additional tools considered in Section 5.13 and the MobSTr dataset have been reviewed and analyzed for the following information:

1. Which artifacts are produced and consumed?
2. Which data formats are used for the artifacts?
3. Is the interpretation of the data formats compatible between producers and consumers of an artifact?

The first question can be answered directly from the engineering scenarios. For answering the other two questions, information from the WP3 deliverables D3.2 and D3.4 is used.

6.1 Coverage Analysis

Figure 6.1 depicts the result of a coverage analysis performed on the engineering scenarios and the activities performed for the MobSTr dataset. It shows an assembled view of the

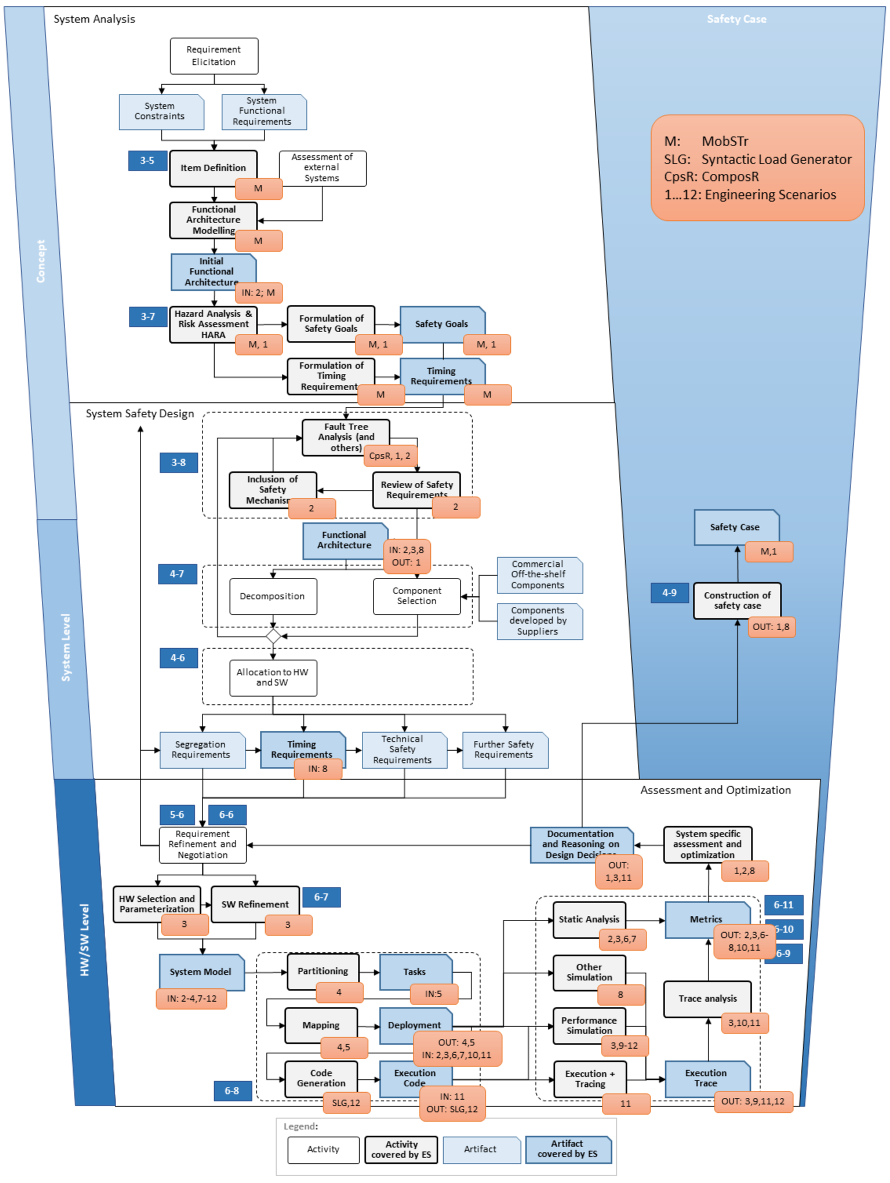


Figure 6.1: Design Steps Coverage

process activities of all three phases as discussed in Chapter 4. In the figure, activities and work products that are covered by at least one engineering scenario are highlighted and labeled with the corresponding source.

In summary, for 21 of 26 (about 80 %) activities there exists at least one engineering scenario where this activity has been considered. The coverage for the work products is lower with 13 of 20 (65 %). This is not surprising as from the requirements side the project was mainly focussing on timing aspects. If we subsume the individual aspects in the *System Safety Design* phase into a single work product, we end up with 13 out of 17, or about 76 %. The figure reveals that the project focus was on the HW/SW design in the *Assessment and Optimization* design phase, with a coverage of nearly 100 %. The only exception is the *Requirement Refinement and Negotiation* activity, which is not explicitly mentioned in the scenarios.

The *System Analysis* phase has been considered mainly in the context of the MobSTr dataset. As the dataset is based on an existing system [RDH+21] with predefined requirements, it does not take the *Requirement Elicitation* activity into account. Neither was an *Assessment of External Systems* intended, as the considered system is self-contained.

The largest gap in the coverage is the interface between system and HW/SW design in the *System Safety Design* phase. These steps were also not in the focus of the project. In the MobSTr dataset, the HW/SW architecture was already defined. This gap is the most serious one with respect to the safety design, as it would have revealed how traceability between the functional and the technical safety concept could be established in terms of the involved requirements. The corresponding clauses in the ISO 26262 are 4-7 and 4-8. The gap is also the most serious one with respect to collaborative design processes, as it is a common interface for the integration of legacy components as well as for components delivered by the suppliers of an OEM.

Table 6.1: Coverage of ISO 26262 Work Products

Part/ Clause	**Work Product**	**Before PANORAMA**	**At the end of PANORAMA**
2-5	Organization-specific Rules and Processes for Functional Safety	-	-
2-5	Evidence of Competence Management	-	-
2-5	Evidence of a Quality Management System	-	-
2-5	Identified Safety Anomaly Reports	-	-
2-6	Impact Analyses at the Item Level	-	-
2-6	Impact Analyses at Element Level	-	✓
2-6	Safety Plan	-	-
2-6	Safety Case	-	✓
2-6	Confirmation Measure Reports	-	-
2-6	Release for Production Report	-	-
3-5	Item Definition	-	-
3-6	Hazard Analysis and Risk Assessment Report	-	✓
3-6	Verification Report of the HARA	-	-
3-7	Functional Safety Concept	-	✓

3-7	Verification Report of the Functional Safety Concept	-	-
4-6	Technical Safety Requirements Specification	-	✓
4-6	System Architectural Design Specification	✓	✓
4-6	Technical Safety Concept	-	✓
4-6	Safety Analysis Report	-	✓
4-6	Specification of Requirements for Production, Operation, Service and Decommissioning	-	-
4-7	Integration and Test Strategy	-	-
4-7	Integration and Test Report	-	-
4-8	Safety Validation Specification including Safety Validation Environment Description	-	-
4-8	Safety Validation Report	-	-
5-6	Hardware Safety Requirements Specification	-	✓
4-6, 5-6	Hardware/Software Interface Specification	✓	✓
5-7	Hardware Design Specification	✓	✓
5-7	Hardware Safety Requirements Verification Report	-	✓
6-6	Software Safety Requirements Specification	-	✓
6-7	Software Architectural Design Specification	✓	✓
6-7	Dependent Failures Analysis Report	-	-
6-8	Software Unit Design Specification	✓	✓
6-8	Software Unit Implementation	-	-
6-9, 6-10, 6-11	Software Verification Specification	-	-
6-6, 6-9, 6-10, 6-11	Software Verification Report	-	-
6-10	Embedded Software	-	-
8-9	Verification Plan	-	-
8-9	Verification Specification	-	-
8-9	Verification Report	-	-
8-12	Software Component Documentation	-	-
8-12	Software Component Qualification Report	-	-
8-12	Software Component Qualification Verification Report	-	-

9-5	Update of Architectural Information	-	✓
9-5	Update of ASIL as Attribute of Safety Requirements and Elements	-	✓
9-6	Update of ASIL as Attribute of Sub-elements of Elements	-	✓
9-7	Dependent Failures Analysis	-	-
9-7	Dependent Failures Analysis Verification Report	-	-

The results of another coverage analysis are depicted in Table 6.1. This coverage analysis focuses on the work products of the ISO 26262:2018 safety standard which can be represented by the PANORAMA approach. As a baseline we use the results of the gap analysis in the Deliverable D4.1 (Gap analysis against ISO 26262) of the AMALTHEA4public project. This gap analysis investigated which of the work products of the clauses of the ISO 26262:2018 standard are covered by the AMALTHEA meta-model (status before the PANORAMA project; third column of Table 6.1). In the PANORAMA project, not only the AMALTHEA but also the ODE meta-model is used to store information during the development life-cycle. Moreover, CAPRA is used to create traceability between AMALTHEA, ODE, and also information provided in other formats (see the MobSTr dataset for details). Hence, there are more work products of the ISO 26262 standard which can be represented with the results of the PANORAMA project (status at the end of the PANORAMA project; fourth column of Table 6.1). While before the project only 5 out of 47 work products could be represented by AMALTHEA, at the end of PANORAMA 18 work products of ISO 26262:2018 can represented by AMALTHEA, ODE, and the traceability provided by CAPRA. So with the work conducted in PANORAMA we could increase the percentage of ISO 26262 work products which are covered in the development life-cycle from 10% to 38%.

Please note that the work products from the clauses 2-x/8-y/9-z in the ISO 26262:2018 standard are part of management or supporting processes and not part of the actual development process itself. Hence, these clauses are not represented in our design process as depicted in Figure 4.1 or Figure 6.1. But for the sake of completeness, we included these clauses in our coverage analysis of the ISO 26262 standard.

6.2 Synthesis

It turns out that three data formats are dominating in PANORAMA: AMALTHEA, BTF, and ODE. Furthermore, C-code can be considered as a format as well, but in contrast to the others it carries a part of the system itself (or an obfuscated variant, in case of the Syntactic Load Generator) instead of a model of the system. These data formats are listed in Table 6.2 together with the artifacts that they represent and the engineering scenarios and tools where they are produced or consumed. The formats collected in the table carry 7 of the 13 covered artifacts (54%), so more than half of the artifacts are exchanged using standardized formats.

Table 6.2: Commonly Used Formats

Format	**Artifact**	**Produced by**	**Consumed by**
ODE	FTA/FMEA	SIEMENS Tool, ES-1, ES-2	ES-2
	Safety Case	ES-1, ES-8	ES-1
AMALTHEA	System Model, Tasks		
	w/o Deployment	–	ES-4, ES-5
	with Deployment	ES-4, ES-5	Syntactic Load Generator, ES-2, ES-3, ES-6 – ES-8, ES-10, ES-11
C-Code	Execution Code	Syntactic Load Generator	ES-11
BTF	Execution Trace	ES-3, ES-9, ES-11, ES-12	ES-12

Table 6.3: Non-Agreed Formats (Not Exhaustive)

Artifact	**Formats**
Item Definition	Plain text
(Initial) Functional Architecture	UML/Eclipse Papyrus, Capella, AMALTHEA?
Safety Goals	Excel
Timing Requirements	Excel, ReqIF, MTSL via ODE, AMALTHEA Constraint Model
Metrics	ATDB, JSON, HTML reports

For some artifacts, no format has been agreed on in PANORAMA. These artifacts and used formats are summarized in Table 6.3.

As already identified in Section 6.1, there is a gap between the tool-supported parts of the process when it comes to functional decomposition and allocation of software functions to hardware. This is because these steps are purely manual in current development practice.

For the analysis of the system safety design coming before the identified gap, a number of approaches and tools have been developed in PANORAMA, including those covered by the engineering scenarios ES-1 and ES-2. For sharing artifacts of this phase, the ODE format is convenient. It is already supported by the FTA/FMEA analysis tools within PANORAMA. Furthermore, the CSW tool [Con20b] has been developed to extract FTA/FMEA information from AMALTHEA models.

For the *Assessment and Optimization* phase following the identified gap, comprehensive tool support has been developed. This is not surprising since this phase has been in the focus of PANORAMA WP3. The tools here can be clustered into five activity groups

Table 6.4: Activity Clusters for Hardware/Software-Level Design

Activities	**Engineering Scenarios & tools**
HW Selection and Parameterization, SW Refinement	ES-3
Partitioning, Mapping	ES-4, ES-5
Code Generation, Execution + Tracing	Syntactic Load Generator, ES-11, ES-12
Static Analysis, Other Simulation, Performance Simulation	ES-2, ES-3, ES-6 – ES-12
Trace Analysis	ES-3, ES-10, ES-11

Safety Case

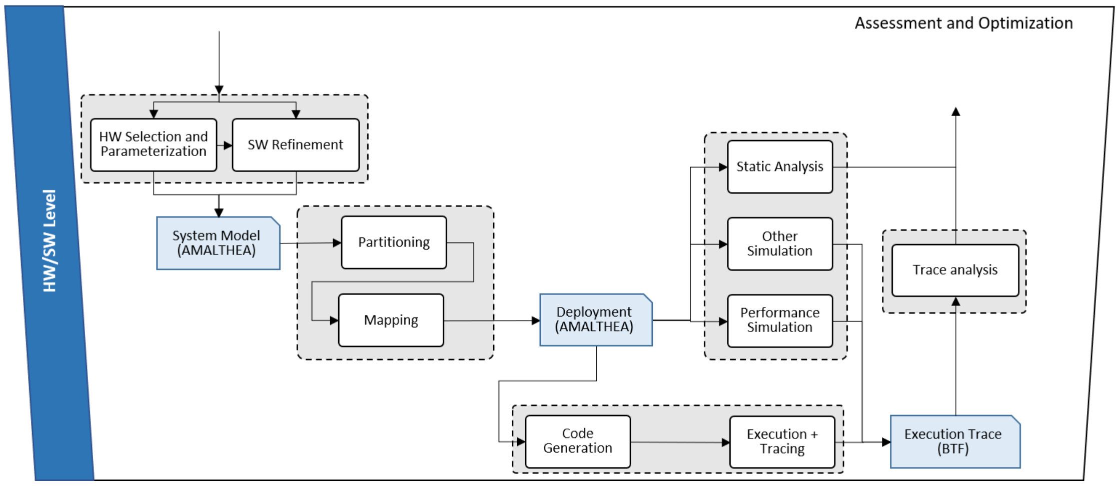


Figure 6.2: Identified clusters and exchange formats between them

listed in Table 6.4 and visualized in Figure 6.2.

Tools covering these activities can exchange information using the AMALTHEA and BTF exchange formats. As a consequence, a comprehensive tool chain can be build by selecting appropriate tools from each of the identified clusters.

For linking different artifacts together (*e.g.*, as part of a safety case), CAPRA is the tool of choice within PANORAMA. Evaluating tracelinks automatically is for example part of engineering scenarios 1 and 8.

In summary, future work is required to establish a common set of exchange formats for the artifacts from the *System Analysis* and *System Safety Design* phases of the design process. Candidate formats have been identified in Table 6.3 and the previous chapters. For the *HW/SW Level*, AMALTHEA and BTF have been strengthened in PANORAMA as common exchange formats that are produced and consumed by a wide range of tools supporting all of the PANORAMA design process activities during that phase. Using these formats, large toolchains can be assembled from both scientific and industrial tools.

6.3 Summary

This report provides valuable results concerning **RQ-4** "How can applicability of the process be ensured?". The engineering scenarios, which are based on the definition of user stories with practical relevance and their implementation exemplify solutions for individual parts of the process. The coverage analysis as well as the synthesis in the previous sections show where and how these parts may fit together.

However, the findings in this report also reveal some open challenges that require further attention. An obvious observation is the missing coverage of activities, which can be roughly split into three groups. First, activities that provide the interface between system level design and HW/SW design are missing. Second, the integration of legacy components and the integration of collaborative organizations, such as in supplier-OEM chains, have not been considered. Last but not least, the definition and the integration of the results of the activities into safety cases have not been well covered. Most of the work at this end has been done in the context of the MobSTr dataset. This was however rather conceptually, and more methodological as well as tool support would be required.

Other gaps are not explicitly visible but are also important in order to instantiate an applicable design process. First, the application of any design process requires well-defined and consistent interfaces. Although discussed in this report, much more comprehensive regulations and definitions are required. Second, collaborative and distributed processes in particular require suitable representation and management of different development states. In software development, revision control systems such as Subversion and Git are devoted to this job. Development processes with their multitude of different interchange formats require similar but typically more complex measures, e.g., to perform diff-management. Strongly connected with the first two challenges is the establishing of traceability. Only a representation of the "design flow" in the work products enables managing complexity and helps reduce the design effort, such as when re-iterations of certain design phases become necessary. In the context of the MobSTr dataset, some traceability aspects and design phases have been elaborated [SKB+21]. A broader consideration of this aspect would be required in order to cover the whole process.

7 Conclusion

This design handbook has summarized the work within the PANORAMA project that focused on design processes and associated topics such as tooling. We have used a number of engineering scenarios from the project partners as well as a prototypical design process that takes common safety standards and, in particular, ISO 26262 into account. The combination between process and engineering scenarios allows us to show that PANORAMA covers a significant portion of ISO 26262, but also helps us to point at the existing gaps, e.g., a lack of support when switching from system-level design to hardware and software design.

An important aspect that we also address are the numerous tools and file formats that are used in the project. While safety standards are important when developing systems in the automotive, avionics, or medical domain, exchange of information and the ability to connect tool chains from different organizations will become increasingly important in the future. We show that the work in PANORAMA contributes towards these challenges. Not only have we increased the coverage of ISO 26262, but we have also demonstrated that exchange formats like AMALTHEA and ODE are useful and sufficient in the engineering scenarios we show.

There is more work to do, of course, and closing the gaps we identified is an open research question. At the same time, the AMALTHEA domain-specific language becomes more powerful and we have demonstrated its ability to interface with many other tools. We therefore believe that a strong and future-proof foundation has been laid on top of which additional tools and methods can close the existing gaps in the future.

Bibliography

[AUT17] AUTOSAR Consortium, *AUTOSAR Virtual Functional Bus Release 4.3.1*, 2017.

[BABB+12] J.-P. Blanquart, J.-M. Astruc, P. Baufreton, J.-L. Boulanger, *et al.*, "Criticality categories across safety standards in different domains," in *Proceedings of the 6th Int. Conf. on Embedded Real Time Software and Systems (ERTS2)*, 2012.

[BBBD+10] P. Baufreton, J. Blanquart, J. Boulanger, H. Delseny, *et al.*, "Multi-domain comparison of safety standards," in *Proceedings of the 5th Int. Conf. on Embedded Real Time Software and Systems (ERTS2)*, 2010.

[BL21] C. Baron and V. Louis, "Towards a continuous certification of safety-critical avionics software," *Computers in Industry*, vol. 125, p. 103 382, 2021.

[Con20a] P. Consortium, "Implementation of dynamic analysis methods," ITEA3 PANORAMA Project, Tech. Rep., 2020.

[Con20b] P. Consortium, "Implementation of static analysis methods," ITEA3 PANORAMA Project, Tech. Rep., 2020.

[Con20c] P. Consortium, "Specification and concepts on system assessment for early design phases," ITEA3 PANORAMA Project, Tech. Rep., 2020.

[Con22] P. Consortium, "Detailed reports about demonstrators and assessment results," ITEA3 PANORAMA Project, Tech. Rep., 2022.

[DWS+10] M. Deng, K. Wuyts, R. Scandariato, B. Preneel, and W. Joosen, "A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements," *Requirements Engineering*, vol. 16, pp. 3–32, 2010.

[GJW+20] M. Gyllenhammar, R. Johansson, F. Warg, D. Chen, H.-M. Heyn, M. Sanfridson, J. Söderberg, A. Thorsén, and S. Ursing, "Towards an operational design domain that supports the safety argumentation of an automated driving system," in *10th European Congress on Embedded Real Time Systems (ERTS 2020)*, 2020.

[HL06] M. Howard and S. Lipner, "The security development lifecycle : SDL, a process for developing demonstrably more secure software," 2006.

[HMS+17] R. Hoettger, H. Mackamul, A. Sailer, J.-P. Steghöfer, and J. Tessmer, "APP4MC: application platform project for multi- and many-core systems," *it Inf. Technol.*, vol. 59, no. 5, pp. 243–251, 2017. DOI: `10.1515/itit-2017-0019`. [Online]. Available: `https://doi.org/10.1515/itit-2017-0019`.

[HMSS16] P. Hohl, J. Münch, K. Schneider, and M. Stupperich, "Forces that prevent agile adoption in the automotive domain," in *International Conference on Product-Focused Software Process Improvement*, Springer, 2016, pp. 468–476.

[HWS17] G. K. Hanssen, G. Wedzinga, and M. Stuip, "An assessment of avionics software development practice: Justifications for an agile development process," in *International Conference on Agile Software Development*, Springer, Cham, 2017, pp. 217–231.

[Int05] Int. Organization for Standardization (ISO), ISO/TC 8/SC 8 Ship design, *ISO 17894:2005 Ships and marine technology – Computer applications – General principles for the development and use of programmable electronic systems in marine applications*, 2005.

[Int18] Int. Organization for Standardization (ISO), *ISO 26262: Road vehicles — Functional safety*, 2018.

[Int98] Int. Electrotechnical Commission (IEC), *IEC 61508: Functional safety of electrical/electronic/programmable electronic safety related systems*, 1998.

[ISO15] ISO, *ISO/IEC 27010:2015 Information technology — Security techniques — Information security management for inter-sector and inter-organizational communications*, Nov. 2015. [Online]. Available: `https://www.iso.org/standard/68427.html`.

[KW04] T. Kelly and R. Weaver, "The goal structuring notation – a safety argument notation," in *Proc. of Dependable Systems and Networks 2004 Workshop on Assurance Cases*, 2004.

[LABB+12] E. Ledinot, J.-M. Astruc, J.-P. Blanquart, P. Baufreton, *et al.*, "A cross-domain comparison of software development assurance standards," in *Proceedings of the 6th Int. Conf. on Embedded Real Time Software and Systems (ERTS2)*, 2012.

[LDvdWB16] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, and S. Brinkkemper, "The use and effectiveness of user stories in practice," in *International working conference on requirements engineering: Foundation for software quality*, Springer, 2016, pp. 205–222.

[MBBB+12] J. Machrouh, J.-P. Blanquart, P. Baufreton, J.-L. Boulanger, *et al.*, "Cross domain comparison of system assurance," in *Proceedings of the 6th Int. Conf. on Embedded Real Time Software and Systems (ERTS2)*, 2012.

[MCA+20] S. Maro, J. Côrte-Real, K. Albers, B. Koopmann, J. S. Becker, D. Schmelter, D. V. R. Sudhakar, M. Bonner, O. Kupriyanov, J.-P. Steghöfer, M. Cruz, and M. Zeller, "Overview of existing development processes," ITEA3 PANORAMA Project, Tech. Rep., 2020. [Online]. Available: `https://itea4.org/community/project/workpackage/document/download/6355/D6.1:%20Overview%20of%20Existing%20Development%20Processes.pdf`.

[PA99] Y. Papadopoulos and J. A McDermid, "The potential for a generic approach to certification of safety critical systems in the transportation sector," *Reliability engineering & system safety*, vol. 63, no. 1, pp. 47–66, 1999.

[PHAB12] K. Pohl, H. Hönninger, R. Achatz, and M. Broy, Eds., *Model-based engineering of embedded systems: The SPES 2020 methodology.* Springer-Verlag Berlin Heidelberg, 2012, p. 304, ISBN: 978-3-642-34613-2. DOI: `10.1007/978-3-642-34614-9`.

[RDH+21] F. Rehm, D. Dasari, A. Hamann, M. Pressler, D. Ziegenbein, J. Seitter, I. Sañudo, N. Capodieci, P. Burgio, and M. Bertogna, "Performance modeling of heterogeneous hw platforms," *Microprocessors and Microsystems*, vol. 87, p. 104 336, 2021, ISSN: 0141-9331. DOI: `https://doi.org/10.1016/j.micpro.2021.104336`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0141933121004956`.

[SKB+21] J.-P. Steghöfer, B. Koopmann, J. S. Becker, I. Stierand, M. Zeller, M. Bonner, D. Schmelter, and S. Maro, "The mobstr dataset - an exemplar for traceability and model-based safety assessment," in *29th IEEE International Requirements Engineering Conference, RE 2021, Notre Dame, IN, USA, September 20-24, 2021*, IEEE, 2021, pp. 444–445. DOI: `10.1109/RE51729.2021.00062`. [Online]. Available: `https://doi.org/10.1109/RE51729.2021.00062`.

[SKHW19] J.-P. Steghöfer, E. Knauss, J. Horkoff, and R. Wohlrab, "Challenges of scaled agile for safety-critical systems," in *Product-Focused Software Process Improvement - 20th International Conference, PROFES 2019, Barcelona, Spain, November 27-29, 2019, Proceedings*, X. Franch, T. Männistö, and S. Martınez-Fernández, Eds., ser. Lecture Notes in Computer Science, vol. 11915, Springer, 2019, pp. 350–366. DOI: `10.1007/978-3-030-35333-9\_26`. [Online]. Available: `https://doi.org/10.1007/978-3-030-35333-9%5C_26`.

[Soc96] Society of Automotive Engineers Inc. (SAE), *ARP 4761: Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*, 1996.

[SSK+21] D. Schmelter, J.-P. Steghöfer, B. Koopmann, H. Mackamul, J. Tessmer, K. Albers, M. Zeller, M. Bonner, M. Ekman, O. Kupriyanov, and R. Weber, “Security concept for distributed collaborative development processes,” ITEA3 PANORAMA Project, Tech. Rep., 2021. [Online]. Available: `https://itea4.org/community/project/workpackage/document/download/7363/D6.3%20Security%20Concept%20for%20Distributed%20Collaborative%20Development%20Processes.pdf`.

[TMSP16] M. Trei, S. Maro, J.-P. Steghöfer, and T. Peikenkamp, “An ISO 26262 Compliant Design Flow and Tool for Automotive Multicore Systems,” in *Proceedings of the 17th International Conference on Product-Focused Software Process Improvement (PROFES’16)*, P. Abrahamsson, A. Jedlitschka, A. Nguyen Duc, M. Felderer, S. Amasaki, and T. Mikkonen, Eds., Springer, 2016, pp. 163–180.