(ITEA 3 – 17003)

PANORAMA

Boosting Design Efficiency for Heterogeneous^3 Systems

Deliverable: D6.1

Overview of Existing Development Processes

Work Package: 6

Design Flow and Traceability

Task: T6.1

Distributed Collaborative Engineering Development Processes

Document Type:	Deliverable	**Classification:**	Public
Document Version:	Final	**Contract Start Date:**	2019-04-01
Document Preparation Date:	Feburary 28, 2020	**Duration:**	2022-03-31

Authors

Salome Maro University of Gothenburg

José Côrte-Real Critical Software

Karsten Albers Inchron GmbH

Björn Koopmann OFFIS

Jan Steffen Becker OFFIS

David Schmelter Fraunhofer IEM

Deepak Vedha Raj Sudhakar Inchron GmbH

Maria Bonner Siemens AG

Olexiy Kupriyanov Siemens AG

Jan-Philipp Steghöfer University of Gothenburg

Magnus Cruz Critical Software

Marc Zeller Siemens AG

Contents

List of Figures

List of Tables

Summary

Collaborative systems engineering processes allow different partners that work together to develop a complex product to synchronise their work and exchange artifacts. While many attempts have been made to describe such processes in theory, practical considerations such as incompatible data formats or complex tool chains impose difficulties. At the same time, the engineering is embedded in a complex legal and regulatory environment.

In this deliverable, we describe the state of the art for collaborative systems engineering processes as well as the state of practice at the companies involved in the PANORAMA project. We also report on a requirements elicitation effort conducted with the involved companies to better understand what the concrete needs for a truly collaborative systems engineering process are. Based on this, we analyse the gaps between the requirements and the state of the practice.

Our work shows that existing processes from DEIS and ARAMIS II address many of the requirements, but there is room for extensions that can be tackled in PANORAMA, in particular w.r.t. tool support for large scale systems, intellectual property protection, integration of information, and living models.

1 Introduction

In large systems engineering projects, thousands of engineers work together to create the end product such as a vehicle or an airplane. This kind of systems development is termed collaborative system engineering, which involves collaboration, communication and coordination between the different individuals, teams and organisations involved in the development to create high quality products [FRMM18].

As system engineering projects grow larger and involve various teams and organisations, adhoc collaborative practices are insufficient and companies have to think about collaboration and plan for it before hand. Yet, there needs to be a balance between formal collaboration and informal collaboration as the later facilitates shorter development cycles [HGK+08]. In distributed environments where the teams are located in different geographical areas, collaboration is crucial but also challenging both due to technical reasons such as tools and non-technical reasons such as, communication and cultural differences [AP10]. Since collaboration happens on different levels such as during project planning, artifacts development and verification and validation activities, organizations need to make sure that the engineers and managers have support for collaboration both in terms of the processes and work-flows in the organizations as well as infrastructure.

1.1 Scope and Objectives

Understanding what and when collaboration activities occur in the systems engineering development process is a first step towards providing process and tool support for such activities. Since the partners of the PANORAMA project work in a highly collaborative environment, we aim to understand how they currently work in order to identify improvement points. The majority of the project partners work in regulated environments where certain standards such as safety and security are relevant. Therefore, while we report on general collaborative systems engineering, this document also focuses on specific aspects such as safety and security and their implication on collaborative development. Therefore this deliverable aims to describe the current state of the art in collaborative software development, how the project partners are currently practising collaborative software development, the challenges they face and the requirements that companies have to improve the state of practice of distributed collaborative development.

1.2 Outline

The remainder of this report is structured as follows. chapter 2 describes how we obtained the data and results reported in this deliverable. chapter 3 describes the state of the art

of distributed collaborative systems development which is derived from predecessors of the PANORAMA project such as DEIS, AMALTHEA4Public and ARAMIS II. chapter 4 describes how the current companies are working in terms of collaborative development while chapter 5 describes their requirements. chapter 6 gives an overview of the gap that exists between the state of the art and the state of practice. The document ends with chapter 7 which gives some conclusions and plans for future work.

2 Methodology

This section describes the methodology we applied to collect and analyze the data in this report. Four main activities which are use case collection, a workshop on requirements elicitation, a focus group on state-of-practice, and a gap analysis that aligns the other activities are described.

2.1 Use Case Collection

The first activity was to collect data on different use cases where collaboration plays a role in systems development. The purpose of this activity was to understand where collaboration is used, which stakeholders, artifacts, tools, and design steps are involved and which purpose these use cases served. We also wanted to capture the multiple points of information exchange between various organizations. In this context, our main goals were thus to:

- identify who collaborates with whom and at which design step of the development process the collaboration happens, and
- identify the information exchanged between the organizations and the infrastructure used during the collaboration process.

For this purpose, we created a use case template, which is shown in Figure 2.1. It was distributed among all PANORAMA project partners to identify the collaborative development processes that are already in place and also to identify the prospective collaborative development processes for the future.

We also created a list (Figure 2.2) to identify the other aspects of the collaborative development processes between various stakeholders. The list is used as an additional input for an interview session conducted among a target group consisting of SAAB, Siemens, Bosch, and INCHRON. The goal of the interview is to elicit the state of practice for collaborative design processes among the target group. Below we summarize the essential collaboration aspects between the different collaboration partners based on the inputs collected from the focus group interview and use-case templates.

2.2 Workshop on Requirements for Collaborative Development

To understand the different collaborative requirements the partners have, we conducted an interactive workshop with 10 participants/9 companies from the PANORAMA project.

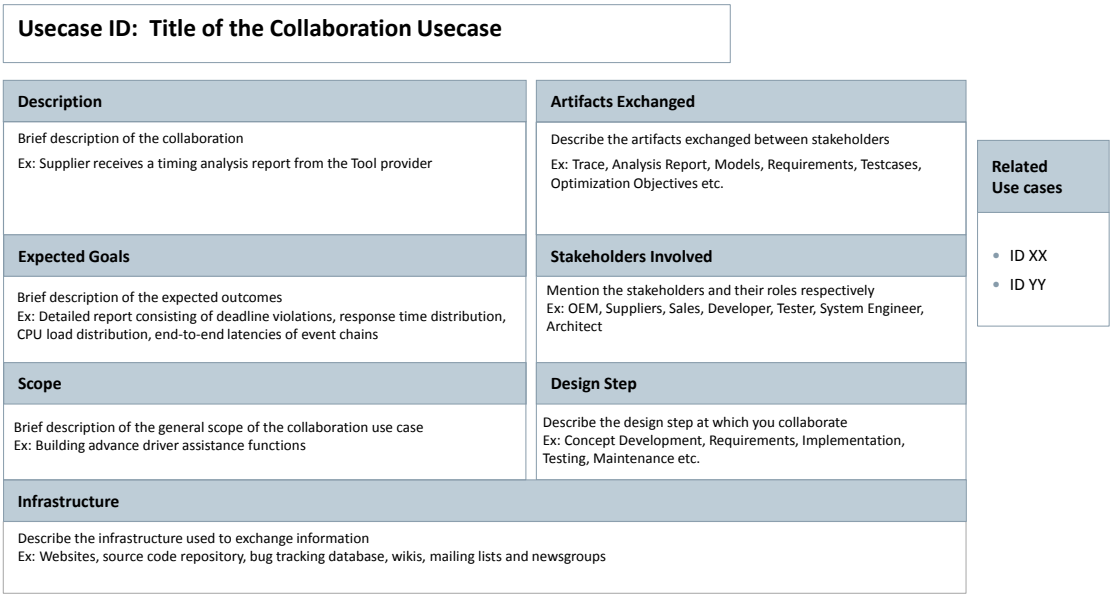

Figure 2.1: Main Use Case Template

During the workshop, participants where divided into 2 groups each with five people. The participants were asked to brainstorm on, organize, and prioritize different ideas on collaborative systems development (cf. figures 2.3 and 2.4). The aim was to come up with as many ideas as possible for how collaboration can be improved.

Ideas Generation: 6-3-5 [Roh69] is a collaborative creativity technique for generating many ideas in a short amount of time. The participants develop/extend three ideas per round on a sticky note. Each round typically takes 3-5 minutes.

Ideas Rating: Magic Estimation is typically used in scrum to get a quick overview about the size of a backlog. We adapted Magic Estimation to rate the generated ideas collaboratively: each participant can increase or lower the priority of a generated idea by moving the appropriate sticky note up or down on a whiteboard silently. After some time, the list of prioritized ideas will stabilize. If certain ideas are moved a lot, those are discussed after the silent phase in order to achieve a common agreement on its priority.

Ideas Organization: Magic Clustering is based on Magic Estimation: Each participant can regroup a generated idea by moving the appropriate sticky note to another group or creating a new group on a whiteboard silently. After some time, the generated groups will stabilize. If certain ideas are moved a lot between groups, those are discussed after the silent phase in order to achieve a common agreement on its group belonging.

The data collected in this workshop was later analyzed to identify concrete requirements for collaboration. The results are presented in section 5.2.

Use case ID	Title of the Collaboration Use case
Preconditions	Conditions that must be true or activities that must be completed prior to executing the use case Ex: Requirement analysis must be completed before starting with implementation
Flow	Describe who exchanges what, alternative flows and exceptions
Postconditions	Describe the state of the system at the conclusion of the use case. Postconditions may include conditions for both successful and unsuccessful execution of the use case.
Traceability	Describe the process, tools, technique to ensure the traceabilty of the artifacts exchanged at this design step
Modeling Framework	Describe the modelling framework used at this design step Ex: SysML, MARTE, EAST-ADL, AADL, MATLAB
Intellectual Property Management	Describe how intellectual property is managed during the collaboration
Viewpoints	Describe the architectural viewpoints of the systems covered by the modeling framework/exchanged artifacts Ex: Logical, Development, Process, Physical View
Guidelines	Describe list of guidelines or safety compliances standards that are met at this design step Ex: Automotive SPICE and CMMI, MISRA, ISO26262, Code generator guidelines.
Data Formats	Describe the list of proprietary or open source data exchange formats used to execute the use case Ex: XML, Amalthea Models, EMF models, Proprietary Models
Tools	Describe the list of tools used at this design step to fulfil the usecase
KPI (Key Performance Indicator)	Describe the list of KPI to quantify the performance of the list of actions performed by the relevant stakeholders
Requirements	Describe any non-functional or special requirements for the system as the use case is executed Ex: Timing, Safety, Reliability, Security etc.,

Figure 2.2: Additional Information for Collaborative Design Process

2.3 Focus Group on State of Practice of Collaborative Development

We conducted a focus group with five companies with different roles in the collaboration process i.e, one OEM, two suppliers and two tool vendors. Our aim was to understand how collaborative systems development happens in practice and from the perspectives of the different companies.

Before the focus group, we prepared a list of questions which were reviewed by the researchers in the work package. The questions explored how the companies currently collaborate in terms of process, artifacts, tools, traceability, safety and security. The focus group followed the interview guide in a semi-structured manner, where the researchers asked questions and the companies answered in a round robin way. The focus group was done online and recorded. Additionally, researchers took extensive notes. The notes were later analyzed by two researchers to identify and describe the state of practice with respect to collaborative software engineering. The results of this workshop are reported in chapter 4.

2.4 Alignment of ISO 26262 with ARAMiS II Generic Process

This section shortly describes the motivation and methodology behind the gap analysis of the AMALTHEA meta-model and the ARAMiS II generic process, as implemented in the ARAMiS II project tool flows [ARA19d], against the requirements from ISO 26262

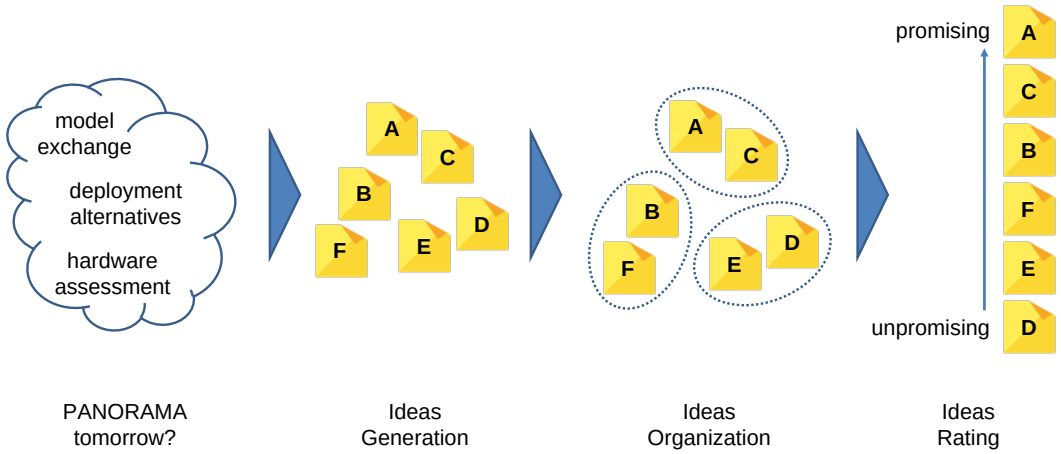

Figure 2.3: Requirements Elicitation Workshop Stockholm: Overview

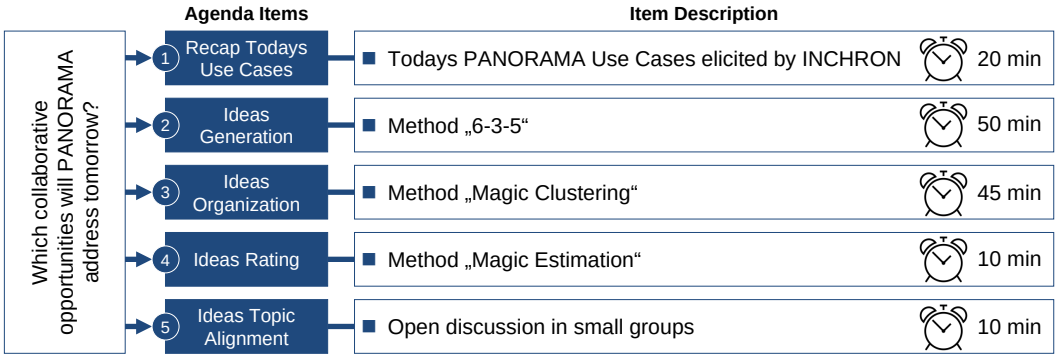

Figure 2.4: Requirements Elicitation Workshop Stockholm: Agenda

[ISO11d] in section 6.1. The project results have been chosen as a reference for the gap analysis because (1) they are a combination of relevant aspects of state-of-practice development processes and relevant safety standards from automotive and aerospace industry and (2) the toolchains developed in ARAMiS II use the AMALTHEA meta-model as a central exchange format along the whole development process. It is also visible from preliminary investigations (see chapter 4) that AMALTHEA indeed is an emerging exchange format within distributed development processes. As a consequence of (1) and (2), the ARAMiS II project results give a good picture that shows how far the AMALTHEA meta-model can support the implementation of safety standards such as ISO 26262.

Since the ARAMiS II generic process cross-cuts development processes from different domains and the focus was on multi-core specific challenges in software development, one cannot expect that the ARAMiS II project results cover all phases of the ISO 26262 safety lifecycle. Therefore, in a first step the clauses of ISO 26262 have been identified that are related to development activities implemented as part of the project. The ARAMiS II

generic process defines input and output artifacts for all these activities. As also ISO 26262 defines input and output artifacts for every clause, the work products from ISO 26262 have been matched to artifacts from the generic process. The description of work products and design steps implemented in ARAMiS II is based on deliverable E2.2 [ARA19b]. In ARAMiS II, also a small dictionary of architecture concepts has been defined that helps matching the artifacts. Based on the mapping for the output artifacts that are results of an activity or clause, respectively, a 1–n mapping from activities to clauses could be defined. The mapping has been confirmed by also matching the input artifacts of the activities and clauses.

In a second step, work products from ISO 26262 have been identified that are result of a matching clause, but do not have a counter part among the artifacts from the generic process. For the other work products – those that have a corresponding generic process artifact – parts of the AMALTHEA meta-model can be identified that are used to represent those artifacts in the ARAMiS II tool chains. The results were extracted from the ARAMiS II project deliverables E2.3 [ARA19c], E2.4 [ARA19d], E3.3 [ARA19e], as well as E5.5 [ARA19f], and widely match earlier results from the AMALTHEA4public project [TMSP16]. As a final step of the gap analysis, the used parts of the AMALTHEA meta-model have been evaluated with respect to the requirements of ISO 26262 on the work products.

3 State-of-the-Art of Collaborative Development Processes

In this section, we provide an overview of the state of the art for collaborative development processes. Thus, we summarise how these processes have been described in the scientific literature and in other projects. After providing a general high-level view of software development processes in section 3.1 and extending it to safety-critical systems in section 3.2, we report on the processes developed in DEIS in section 3.4 and ARAMIS II in section 3.5. Finally, we also introduce the general workflow supported by Siemens Polarion in section 3.6.

3.1 Software Development Standard Process

In practice, every development process is adaptable, depending on the system context. However, the process generally follows the life cycle of Figure 3.1, which can cover most software development projects on heterogeneous systems:

System Engineering consists in the initial specification of the system/software to be developed including the features list, architectural design (high-level design) and – if applicable - first user interaction solution previews (form and behaviour).

- **Kick-off Meeting (KOM)** is the formalization of the beginning of the project. This usually has as input the initial version of the Technical Specification contains a high-level definition of the features / functions, main components, which may include User Interfaces (Mock-ups), and high-level solution and design of the system to be developed.

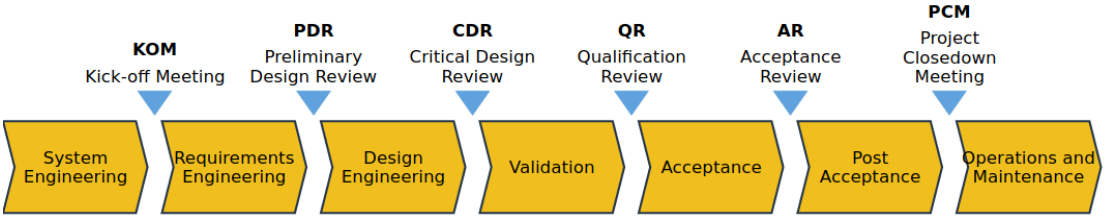

Figure 3.1: Critical Software – Software Development Generic Life Cycle

Requirements Engineering consists in the specification of the software to be developed including the software requirements, architectural design (high-level design) and user interaction solution previews (form and behaviour).

- The software requirements engineering phase is completed by the **Preliminary Design Review (PDR)**. The inputs to this milestone are: the Software Requirements Specification, system architecture, the preliminary interface control document and Solution Previews (e.g. Mock-ups, lo-fi and/or hi-fi Prototypes), all part of the Technical Specification.

Design Engineering produces the detailed design and source code in parallel allows the design to be generated from source code using reengineering tools. Producing the unit testing in parallel with the coding allows the errors to be identified and corrected earlier.

- The results of this phase are the input to the **Critical Design Review (CDR)**, which signals the end of the design phase. The state of the software project after critical design review is called Defined State.

Validation verifies the end-to-end functionality of the system in satisfying all requirements and specifications (mainly system testing), including system usability verifications.

- The validation phase includes a **Qualification Review (QR)**. The state of the software project after qualification review is called qualified state.

Acceptance demonstrates that the system meets your requirements in the operating environment through testing conducted under the supervision of an independent acceptance testing team and follows the procedures specified in the Acceptance Test Plan.

- The acceptance phase includes an **Acceptance Review (AR)**. The state of the software project after acceptance review is called the accepted state.

Operations and Maintenance process is activated when the software product undergoes any modification to code or associated documentation as a result of correcting an error, a problem or implementing an improvement or adaptation.

3.1.1 PANORAMA Context

The PANORAMA project aims to provide modeling tools that will support mainly the design engineering phase and the validation phase. For instance, we aim to support modeling of safety related models such as Faults Tree Analysis (FTAs) models and Failure Modes and Effects Analysis (FMEA) models. We also aim to facilitate the use of analysis results from the different analysis tools for improvement of the models. Since various modeling languages are used in the different steps, in the PANORAMA project aims to extend the AMALTHEA model so that it can be integrated with the various commonly used modeling tools to allow for smoother collaborative work.

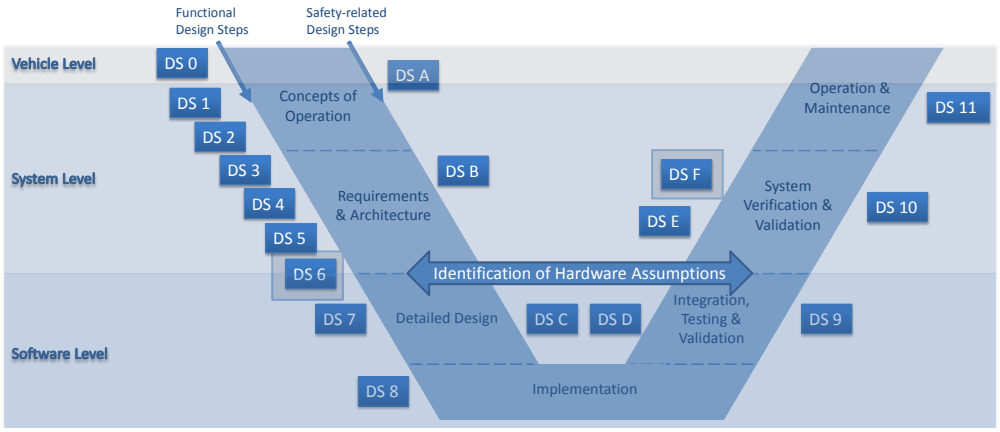

Figure 3.2: Safety-critical Systems Development Life Cycle [TMSP16]

3.2 Safety-critical Systems Development Process

In section 3.1 we have described a generic process for software development. In this section, we describe a typical development design flow for safety critical systems that includes both hardware and software [TMSP16]. This process describes the functional steps as well as safety-related steps required to design a whole system. This design flow is a result from the project AMALTHEA4Public, and is compatible with the ISO 26262 standard. It is depicted in Figure 3.2. Step **DS 0** to **DS 11** are functional design steps while steps **DS A** to **DS F** are safety related design steps. A summary of these design steps is given in Table 3.1. Additionally, more details on how this fits into collaborative systems engineering process are given in section 5.1

3.3 Collaborative Work in Tool Platforms

The development of heterogeneous embedded systems involves coming together of different organizations with expertise in different domains. Realizing such systems requires concrete collaboration between specific groups of different organizations. Hence for efficient collaboration between different organizations, interoperable tool support is necessary. Essential aspects of collaborative work in tool platforms may include data management, information management, data privacy, data security, and tool usability [CrE19].

Emails are one of the most commonly used communication tools to share information among organizations. However, emails may not be well suited for specific tasks such as working on a draft that involves the input of multiple parties. Essential information such as comments, context information, and managing of documents are challenging to track and can get mixed up in a long chain of email history. Version control systems (VCS) improve the document editing process, as every partner has a view of current state and complete version history, including tags for special versions. The VCS such as

Table 3.1: Overview of the Identified Design Steps

Functional Design Steps	
DS 1: System Requirements Engineering	DS 7: Variant Configuration
DS 2: System Architecture Design	DS 8: Implementation
DS 3: Software Requirements Engineering	DS 9: Validation and Testing
DS 4: Derivation of Product Variants	DS 10: System Integration
DS 5: Definition of Software Architecture	DS 11: Handover
DS 6: Behaviour Modelling	
Safety-related Design Steps	
DS A: Derivation of the Functional Safety Concept	DS B: System Safety Requirements Engineering
DS C: Software Safety Requirements Engineering	DS D: Verification of Software Safety Requirements
DS E: Safety Validation	DS F: Functional Safety Assessment

GIT [Git19], SVN [Apa19] also offers integrated diff tool that highlights the differences between different versions. The VCS also provides the possibility to resolve conflicts before a new version is created. Although VCS offers a better solution to manage data; however, context information such as task sharing among partners needs to be tracked using a separate tool such as kanban board.

3.3.1 Document-centric Collaboration

Document-centric collaboration tools keep track of the collaboration metadata in addition to the document. Comments (metadata) created are highlighted in the document and are visible to everybody. Hence the problem of missing context information is addressed by keeping track of the metadata. Cloud collaboration tools share the data in a public cloud. Therefore, documents can be shared with external organizations by taking appropriate security measures. The editing possibilities of the documents are supported by desktop and web applications. The web services offered by cloud collaboration tools enable interoperability with other web services using public APIs. Real-time collaboration tools are a special type of cloud collaboration solution that allows multiple users to work together at the same time. Ad hoc discussions can be started with the help of integrated chat and video conferencing capabilities. Hence, real-time collaboration tools enable all collaborators to see the changes made by each other instantaneously and offer the possibility to react immediately or later.

3.3.2 Artifact-centric Collaboration

Collaborative development between different organizations not only involves sharing of documents but also involves sharing of other artifacts such as models. The artifact-centric

collaboration process facilitates the exchange of well-defined artifacts between partner organizations. Essential aspects of artifact-centric collaboration tools are described as follows [CrE19]:

- Supports synchronization of the artifact data such that all collaborators can see the updated version as soon as possible.
- Avoids introducing new bugs to the artifacts during the process of editing to ensure the safety of the artifact exchanged.
- Enables automatic notifications of artifact changes to all the partners involved.
- Enables ad-hoc communication between different partners.
- Supports version management of artifacts.
- Enable legacy support for importing and exporting artifacts

3.4 Distributed Dependable Systems Development

The following collaboration scenarios were specified as part of the H2020 research project DEIS[1]. In industrial practice, the development of complex, safety-critical systems is scattered across different partners (e.g., OEM, Tier-1 suppliers). This is the case for instance in automotive, avionics, or railway domain. Thereby, the suppliers need to take over an increasing share of the risk from the OEM. Moreover, the OEMs builds their safety case based on the information provided by the suppliers.

A high amount of alignment activities is required for ubiquitous feature development and several iterations are needed to align the interdependent function development. Moreover, lots of assumptions and constraints for development of elements-out-of-context are made by the suppliers. Hence, interdependent functions are avoided as much as possible, therefore innovative functionalities are hampered. All partners involved in the development process interchange safety- but also reliability-related information. Thereby, all involved participants along the supply chain use different methodologies and tools for engineering functionalities. Today, the information is exchanged using documents.

3.4.1 Collaboration Scenario: Requirement-driven Design

The OEM provides the safety requirements which must be taken into account by the suppliers. The individual component suppliers can base their assumptions on the exchanged information and update the context their sub-system/component. The suppliers themselves provide safety information (e.g., about the conducted safety analyses as well as their safety concept) in addition to the component/sub-system they deliver to the OEM (see Figure 3.3). This information is exchanged with other suppliers and the OEM (during the integration stages of the development life-cycle). Based on this the OEM is able to generate a safety case for the target product.

[1] http://deis-project.eu/

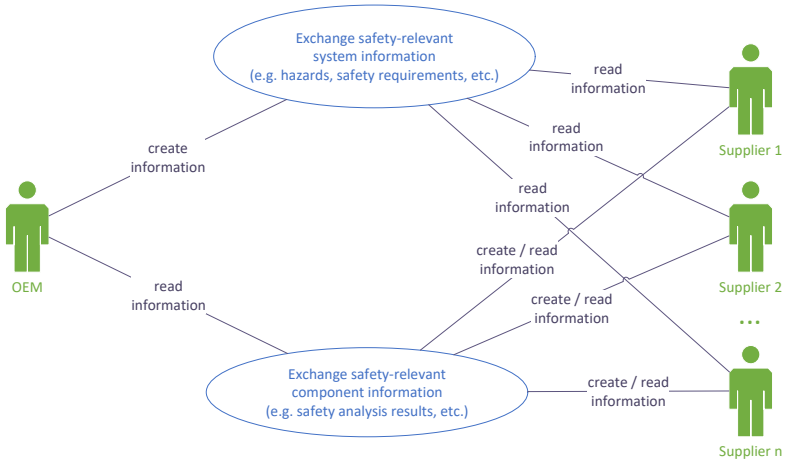

Figure 3.3: DEIS Collaboration Scenario 1

3.4.2 Collaboration Scenario: Components-of-the-Shelf

The OEM builds its system based on pre-existing components of the suppliers. Thereby, the OEM does not provide safety requirements to the suppliers. However, the suppliers need to deliver safety information to the OEM in addition to the component/sub-system (see Figure 3.4). Based on this information the OEM builds its safety case for the overall system.

3.4.3 Collaboration Scenario: System-of-systems Integration

The OEM provides all necessary dependability information related to a product/system which must be integrated into a (pre-existing) system-of-system (see Figure 3.5). Hence, the system-of-system operator can create an safety case for its system-of-system.

3.4.4 Challenges

Seamless interchange of safety information enables the creation of safety cases and/or the assurance of correct integration for systems/products. Moreover, the safety requirements provided by the OEM helps the supplier to develop a dependable sub-system/component. Since safety-related information are currently exchanged in a document-based way, it is too much effort for the involved parties to read and understand the information and to enter the data into the own custom tool chains. A machine-readable format is required to formalize the information and to automate the exchange. Moreover, since different methodologies and tools are used by the partners along the supply chain for the various engineering activities, a tool-independent exchange format is required.

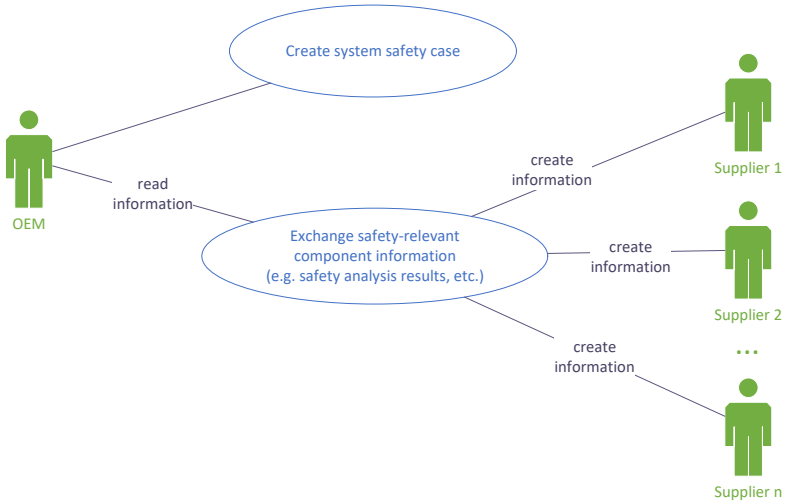

Figure 3.4: DEIS Collaboration Scenario 2

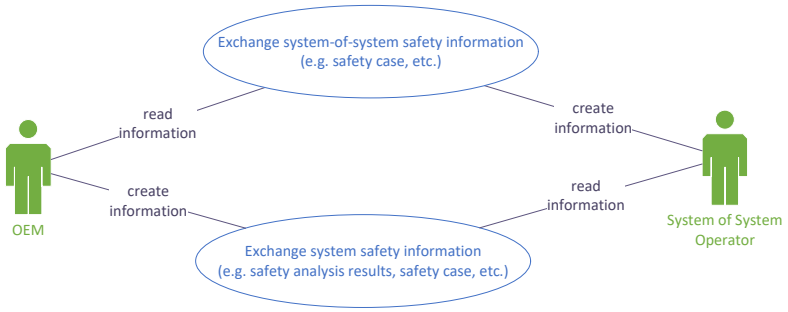

Figure 3.5: DEIS Collaboration Scenario 3

3.5 ARAMiS II Generic Process

Within the ARAMiS II project [ARA19a], a common and generic design process that covers a variety of structured development processes used in industrial practice of multiple domains such as automotive, avionics, and industrial automation has been defined. It includes comprehensive expertise and know-how from manufacturers (e.g., Audi, Bosch, Continental, Denso, Airbus, Diehl, Liebherr, Siemens, Wika), research institutes (e.g., DLR, OFFIS, Fraunhofer, KIT, fortiss), and tool providers (e.g., Timing Architects, Vector, AbsInt, Symtavision, Silexica).

While the focus is on the development of multicore systems, it is based on the commonly used V-model as a state-of-the-art process model. In addition, the generic design process is aligned with a variety of international standards such as *Functional Safety for Road Vehicles* (ISO 26262), *Software Considerations in Airborne Systems and Equipment Certification* (DO-178C), *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems* (IEC 61508), *Systems and Software Engineering – System Life*

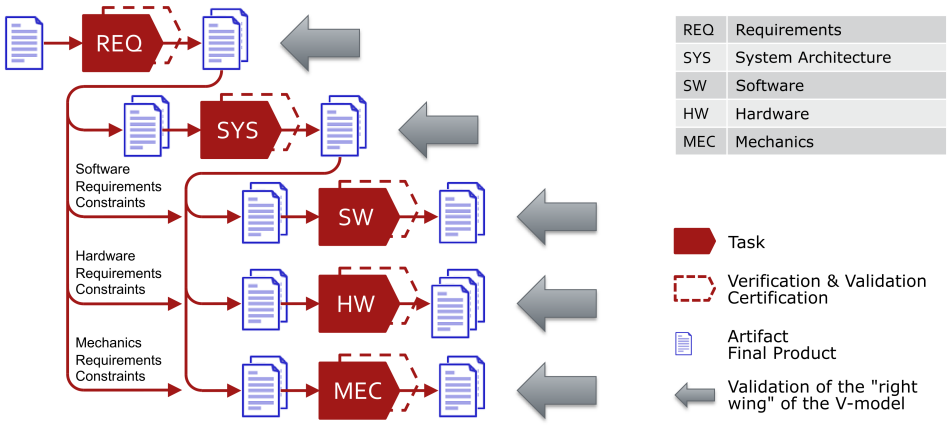

Figure 3.6: ARAMiS II Generic Process [KS19]

Cycle Process (ISO 15288), and *Systems and Software Engineering – Software Life Cycle* (ISO 12207).

The description of the generic design process consists of tasks/activities and corresponding work products. The individual parts are defined in terms of the *Software Process Engineering Metamodel* (SPEM). The basic process is divided into five key activities, which are depicted in Figure 3.6 and will be examined more closely in the following sections. Work products are those that have been identified to be relevant among all industrial project partners.

3.5.1 User and System Requirements Engineering

As a first step, user and customer needs are collected in the *User and System Requirements Engineering* (REQ) activity. Moreover, the problems to be solved are formulated as a set of requirements. Subsequently, these requirements are transformed into system requirements that have to be fulfilled by the system under development.

Inputs Customer Needs

Outputs System Definition and Interfaces, System Requirements and Constraints

3.5.2 System Architecture

Based on these first results, the analysis of the system requirements is carried out in the *System Architecture* (SYS) activity. Following the design of the system architecture, which identifies the system elements and their relationship as well as the mapping between

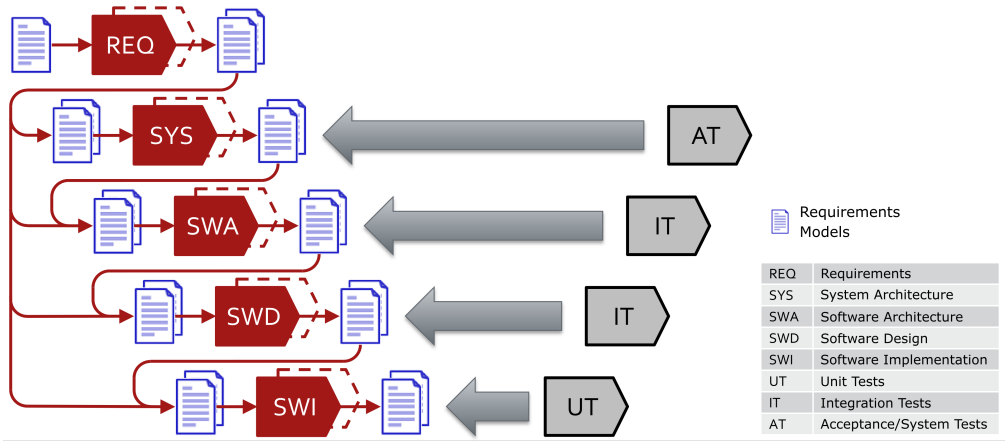

Figure 3.7: ARAMiS II Generic Software Development Process [KS19]

requirements and elements, it is decided which elements are realized by hardware, software, and/or mechanics.

Inputs System Definition and Interfaces, System Requirements and Constraints

Outputs System Architecture, Software Architecture Requirements and Constraints

3.5.3 Software Development

The *Software Development* (SW) activity comprises the whole software development process, which is summarized in Figure 3.7. Besides the analysis of the system architecture and the development of the software architecture in the *Software Architecture* (SWA) activity, this step also includes all *Software Design* (SWD) and *Software Implementation* (SWI) activities.

3.5.4 Hardware Development

In analogy to the SW activity, the *Hardware Development* (HW) covers the whole hardware development process, which is illustrated in Figure 3.8. It is subdivided into the *Hardware Architecture* (HWA), *Hardware Design* (HWD), and *Hardware Implementation* (HWI) activities.

3.5.5 Mechanics Development

The last step to be mentioned is the *Mechanics Development* (MEC) activity.

3.5.6 Verification and Validation

As indicated in Figures 3.7 and 3.8, the corresponding verification and validation steps are performed along the individual tasks. Besides the elicitation of relevant process phases a

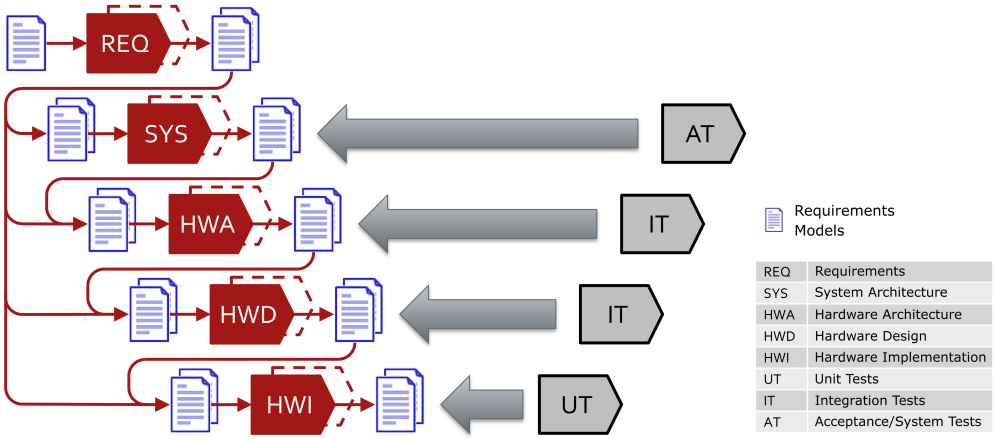

Figure 3.8: ARAMiS II Generic Hardware Development Process [KS19]

common structured terminology has been defined.

3.5.7 Importance for PANORAMA

In the context of PANORAMA, the generic design process will be used as a tool to structure the requirement elicitation process and to align the design process with all standards mentioned above. It is important to note that the process does not cover all aspects of the development process. Just to name a few examples from ISO 26262, missing things are the planning of safety activities such as validation plans (clause 4-6), integration and testing plans and specifications (clause 4-8) as well as verification plans and specifications (clauses 6-6, 6-9, 6-10, 6-11).

3.6 Collaboration Traceability Workflow

This section describes how the Polarion platform [Sieb] allows to find an effective way to organize the collaborative process across different teams and to manage multiple projects along the development process stages depending on the project specifics [Sie16a].

Web-based Collaboration The Polarion platform is the browser-based front-end. Thanks to the always-up-to-date online environment with live dashboards as well as access-controlled threaded commenting the tool is designed to eliminate emails and meetings in order to keep all the information on specification, change requests, design decisions and etc. in one system. One centralized repository at the core of all activities serves the single source of truth. The centralized nature of information exchange enables development teams sitting in different locations to effectively convey ideas, much faster than the use of email, instant messaging and teleconference.

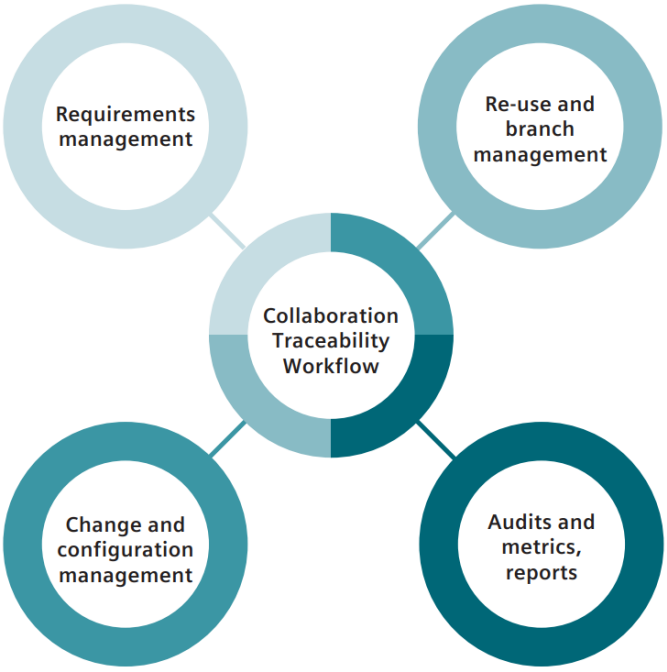

Figure 3.9: Key Features of the Polarion Platform

Team Collaboration Stakeholders can collaborate and communicate on various levels. For discussions and collaboration at higher levels, Polarion features a built-in wiki with default wiki spaces and documents for the repository and each project. This provides a highly flexible communication medium accessible to everyone with access rights to the repository, project or document.

More granular collaboration and communication takes place in comments on individual work items. Discussions on multiple threads can occur among project team members. Comment visibility can be optionally controlled and limited; for example, some comments may be visible only to managers.

Interchange between OEMs and Suppliers Domain experts who want to stay in their familiar environments can do so and still be tied into the centralized repository. The Polarion software's native integration with MATLAB®, for example, enables customers to include Simulink Model-Based Design workflows as an integral part of their application lifecycle. Bidirectional traceability facilitates navigation from Simulink model elements to associated Polarion work items and vice versa. Versioning aids collaborative design, opening up the assets for easy re-use and variant management across an entire automotive portfolio.

Another native integration that is popular among automotive customers is the round-trip for Requirements Interchange Format (RIF/ReqIF) through which traceability across multiple documents or tools is maintained. The Object Management Group's (OMG's)

standard for requirements exchange, a widely used Extensible Markup Language (XML) file format and workflow to support lossless exchange between partners, brings OEMs and their suppliers together around the globe.

Sharing and Reviewing Documents/Work Products Polarion also enables data modification, including approval of requirements via Word documents. Using the Polarion unique Round-trip for Word capability, documents containing managed artifacts can be exported to a Word document, which can then be shared with and reviewed by people who don't have access to Polarion. After changes (the type of which can be optionally restricted during export), the Word document can be re-imported to Polarion, where the changes it contains are incorporated into the online document, and the document history is updated.

Development Workflow Polarion allows to establish a workflow among diverse groups within and outside the organization working together. A customized solution can be established in the beginning, or a template for most common methodologies, as for example, V-Model, Agile Software Project, can be chosen and configured to map to specific business scenarios.

Traceability Comprehensive traceability allows developers to refer back to the software requirements that underlie their assigned tasks, and to reach out to the respective authors when they have questions. The same applies to the testers that verify whether the requirements have been met. All activities and decisions are automatically tracked, with collaboration history available to reveal how decisions were made every step along the way. Formal approval processes with compliant e-signatures complete the information exchange.

ISO 26262/IEC 61508 Qualification by TÜV NORD Siemens PLM Software is the first ALM vendor to receive ISO 26262/IEC 61508 qualification by TÜV NORD for the Polarion suite of products. The qualification at the highest Automotive Safety Integrity Level (ASIL-D) as defined in ISO 26262 is based on evidence that Polarion's software development processes can be reliably implemented and replicated. Due to the nature of the qualification, any software and hardware systems developed using Polarion's processes is also deemed to meet the functional safety requirements of ISO 26262, in turn radically reducing compliance efforts.

Future Goal within PANORAMA Project As it was mentioned, the Polarion platform allows to organize a workflow and to customize the traceability links according to a specific methodology by defining an extension[Siea]. Such an extension would allow Polarion customers to arrange a collaboration process in a faster and in a more effective way. It is planned that an extension established within the PANORAMA Project covers a collaborative workflow for timing analysis compliant with PANORAMA model (support timing requirements and constraints using compatible information models).

A future PANORAMA extension would involve template specification, which is basically definition of semantics of the basic Polarion elements. As for example, work item plays a role of a basic element and can be related to anything you want to track in the project [Sie16b]. Regularly, a work item turns into requirements, activities, change request and test cases. The relationship between working items can be customized as well. Continuing this way, the Polarion platform gives a lot of opportunities to customize a working flow. Additionally, it is planned to integrate with architecture modeling tools, such as Mentor Capital Systems and Mentor Capital Software Designer, where bidirectional traceability across requirements/tests/risks/etc. and modeling elements is already implemented.

4 State-of-Practice of Collaborative Development Processes

In this chapter, we present the combined results of the data collected with the use case templates and with the focus group (cf. chapter 2).

4.1 Collaboration Workflow

Typical collaboration relationships are between OEMs and suppliers as well as suppliers of different tiers. Tool providers collaborate with customers on all levels of the supply chain. If collaboration takes place within a supply chain, e.g., for a vehicle or an aircraft, there are different approaches to managing the contracts. Either the OEM sub-contracts development to a Tier-1 supplier, which in turn sub-contracts to Tier-2 suppliers, etc. or the OEM coordinates the collaboration with appropriate contracts all the way down the supply chain. While this depends on the type of project, in practice, it also depends on the existing relationship between the partners. The current relationship also determines the level of formality of the contract, which can range from a loose general agreement to a full-fledged formal contract between partners.

In most cases, the collaboration between the partners is ongoing throughout the entire development process with differing intensity. Collaboration is most intense when the project ramps up, and the specifications are created and discussed, and the contracts are defined. Later, collaboration is most intense whenever changes occur. Such changes, e.g., in the specification, may require conflict resolution on both the technical and contractual level and can even require a renegotiation of the contract. Many collaborations also showcase higher intensity when the solution is tested. During "normal" development, interactions are usually restricted to clarifications. At the early stages of collaboration the partners meet more often in a face to face meeting and conduct workshops to facilitate the collaboration process. As the collaboration matures over time, video and audio conferencing tools are used to track the progress.

How the partners interact is specified in different ways and with varying levels of formality. In some cases, interactions are defined in the contract, in particular in case of interactions that have an impact on the contract itself. Contracts also include deliverables, schedules, and sometimes artifact formats. The latter can also be specified in separate documents or can even be left in the form of an informal agreement between the involved engineers. In the case of regulated environments, activities, and artifacts that require collaboration, need to be introduced due to standards such as ISO 26262 or DO-178B which both require constructing safety cases that include components delivered by suppliers.

Contracts can also be accompanied by more detailed operationalizations of the general terms in project planning documents or even wikis that describe the process.

Roles that are typically involved in managing the collaborative effort are project managers and team leaders, technical managers, and business development managers. Depending on the project, engineers often have direct interaction with each other. In some cases, a dedicated engineer is co-located with the collaboration partners or directly responsible for handling a specific customer. In terms of the workflow, the processes of the partners can differ, but there are usually synchronization points. These are defined in terms of stages, gates, or milestones during the contracting or project planning phase. A full synchronization, e.g., in terms of sprint lengths, joint reviews, etc., can also take place but is usually restricted to agile projects.

4.2 Artifacts Exchange

Some of the most common collaboration goals in the context of the panorama project involve efficient design space exploration of the system to make early design decisions. Collaboration partners exchange a variety of artifacts to achieve their collaboration goals. Most notably, they exchange requirements documents. They often include natural language requirements and are formatted as spreadsheets or are simple text files. In later stages, models are exchanged, e.g., to be able to simulate the behavior of a component. Such simulation models are, e.g., in MATLAB/Simulink, EAST-ADL, Amalthea, or dSpace format. Suppliers can also provide base software, i.e., software that has not been customized for the specific usage scenario to the OEM for preliminary testing and exploration. Tool developers also receive process descriptions from their customers used to customize the tools. In turn, they deliver step-by-step instructions on how to integrate the new tooling into the process. It is also not uncommon for a customer to share analysis results or raw data for analysis with the provider of an analysis tool to simplify the development of a new feature. Very often, such information is also delivered in a spreadsheet. The analysis results and the reports generated from the tools also play an essential role in the certification process of OEM's.

4.3 Infrastructure

Depending on the existing relationship and the sensitivity of the information, exchange of the artifacts can occur on physical storage media, via email, a shared folder, or a shared repository. Suppliers, in particular, need to set up an infrastructure to process these artifacts. In the case of large suppliers with a steady relationship with OEMs and sub-suppliers, toolchains can be developed specifically to deal with the individually agreed on exchange formats. Tools like IBM Rational DOORS, Enterprise Architect, Kanban, or Siemens Polarion allow a range of import formats. However, sometimes the exchange formats are proprietary and need to be converted. If such exchanges occur repeatedly, the partners invest in developing specialized tools for this purpose. Otherwise, the conversion

is done manually.

4.4 Traceability

In order to achieve consistency, in particular, when updates occur, unique identifiers are often associated with the different elements contained in the exchanged artifacts. It allows identifying, e.g., a requirement even in a different version of the requirement specification, and provides the ability to create traceability links that survive an update of the documents. For many projects, especially safety-critical projects, traceability links are also part of the exchanged artifacts and not delivered as separate artifacts. The form of these links ranges from simple ID tags, e.g., a requirement ID in the code comments to hyperlinks that can be navigated for cases where tools like Siemens Polarion are used by both partners. The main use of the traceability links is for certification purposes; however, sometimes the links are used to inspect if every requirement has been met during the delivery of components and to discover the causes of defects when defects are detected. Configuration management tools can also help in tracking different versions and identifying and resolving conflicts.

4.5 Security and Intellectual Property Management

In terms of *information security*, it is not uncommon that sensitive information needs to be redacted from an artifact before it is exchanged. This can include information about the function of a model, the actual functionality and algorithmic details, performance parameters, data accuracy, and sensor rates. The most common reasons for this are the need to protect intellectual property and regulation, in particular in the defense sector. These needs are usually captured in the initial contract and assured throughout by, e.g., creating different models of the same component with different level of detail, hiding information by replacing identifiers with placeholders, imposing strict access control on sensitive information, or encrypting information when exchanging it via insecure channels. In some cases, it is also possible to exchange binaries instead of source code or create special "opaque" models that cannot be introspected.

Although in many cases information protection is defined on an ad-hoc basis, there is a growing need for companies to embrace new regulations and certifications that aim to harmonize how security risks are handled in an organization. ISO/IEC 27001 is an information security standard that defines well established requirements for an organization to implement an Information Security Management System (ISMS), that serves as a central authority to all matters related to information security, regardless if the information is contained in IT systems or hard-copy documents. This standard places information security as a top level management concern, provides tools to measure and mitigate security related risks, and creates a common ground to enable accredited certification bodies to audit an organization's ISMS and grant them a proof of certification.

5 Requirements of Future Collaborative Development Processes

We collected requirements from two different sources: ISO 26262, the safety standard used for vehicles, and an ideation workshop we conducted with the project members. The requirements from ISO 26262 on the process, the models, and the validation and verification tools are described in section 5.1. The requirements provided by the project partners are listed as user stories in section 5.2.

5.1 Requirements from ISO 26262

The international standard ISO 26262 [ISO11d] defines a safety lifecycle for automotive applications, i.e., a process to be performed during development of safety-critical automotive applications. Besides the safety lifecycle, ISO 26262 defines requirements on activities supporting the development process, such as change and configuration management, and documentation.

5.1.1 Overview on the Safety Lifecycle

The safety lifecycle defined in ISO 26262 is a classical V-process. It starts with a hazard and risk analysis (HARA) of the system being developed. For each identified hazard, a set of top-level safety requirements (safety goals) are defined for the system. During the development phase (left side of the V), the system is decomposed into hardware and software elements. The process is requirements-driven, so safety requirements for each element are derived from the safety goals. ISO 26262 covers both hardware and software development; on the hardware part, an important goal is to ensure that the system can cope with random hardware faults. For this, ISO 26262 defines target values for so-called *hardware architectural metrics* [ISO11a] that need to be evaluated during development.

ISO 26262 defines a set of safety activities to be carried out before and during the development of the system. They are sketched in Figure 5.1. At begin of the process, a *safety plan* is created. In each development phase, analysis and verification activities are planned as part of the safety plan. During the design phase, correct decomposition of components and requirements is verified, and during the integration and testing phase verification ensures correct and complete implementation of the safety requirements. After integration, the *safety validation* provides evidence that the complete system meets the safety goals. So-called confirmation reviews ensure the correct and complete execution of the analysis and verification activities. During the safety lifecycle, the results of the activities are used to form the *safety case*, i.e., to provide an argument that the system

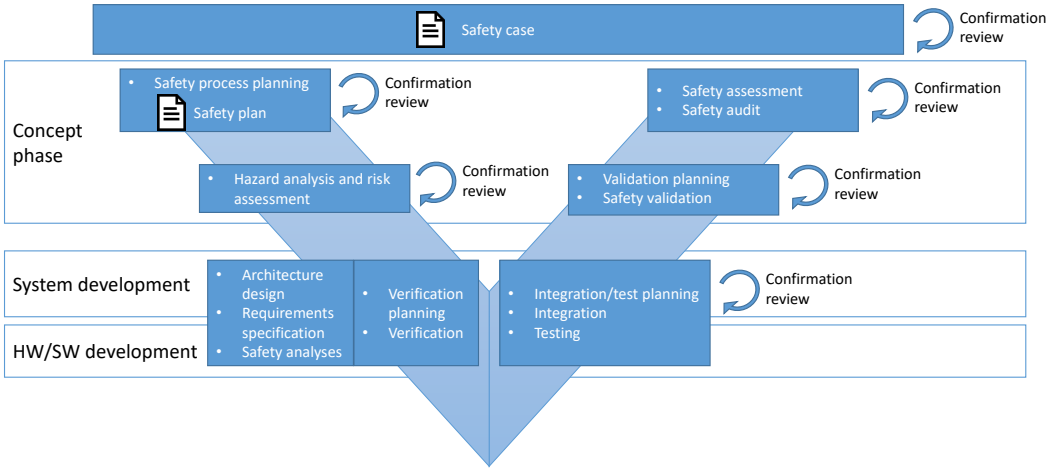

Figure 5.1: ISO 26262 Safety Activities

meets the safety goals [BRH+13]. In the *safety assessment* and *safety audit* at the end of the safety lifecycle, all the activities are evaluated and a judgment is made whether safety of the system has been achieved.

5.1.2 Automotive Safety Integrity Levels

In the ISO 26262 safety lifecycle, architecture elements (e.g., software components and hardware parts) and safety requirements are equipped with an Automotive Safety Integrity Level (ASIL). The ASIL ranges from A to D, with A the lowest and D the highest level. A special value QM is assigned to elements that are not safety critical and therefore outside the scope of ISO 26262. The higher the ASIL, the more stringent are the requirements of ISO 26262 on development of the element. For the safety goals, the assigned ASIL depends on the probability and severity of possible accidents in case of the hazards that are prevented by that safety goal.

In general, new architecture elements and requirements inherit the ASIL of their parent, i.e., the safety goal they belong to. However, ISO 26262 allows to decompose components into components with lower ASIL, provided they are sufficiently independent and together implement some kind of redundancy (i.e., the parent component fails only if multiple of the children fail). This is named ASIL tailoring in ISO 26262 [ISO11c; WC12]. In case ASIL tailoring is applied, the ASIL of sub-components must sum up to the ASIL of the parent. Decomposed components and their safety requirements have extended ASIL assignments in the form $X\,(Y)$ (with $X, Y \in \{QM, A, B, C, D\}$) where X is the components reduced ASIL and Y the initial ASIL of the safety goal where the components have been derived from. The inherited ASIL is the same among all levels of the design. A component with ASIL $X\,(Y)$ may thus be decomposed into two components with ASIL $X_1\,(Y)$ and $X_2\,(Y)$ such that $f(X_1) + f(X_2) \geq f(X)$ with $f = \{QM \mapsto 0, A \mapsto 1, B \mapsto 2, C \mapsto 3, D \mapsto 4\}$ [FVG19]. The reference to the initial ASIL is important because some properties of

the components (for example hardware architectural metrics) are not subject to ASIL tailoring and still be evaluated according to the ASIL of the safety goal.

5.1.3 Work Products and Documentation

ISO 26262 defines a large set of about 100 work products that are the result of the activities required by ISO 26262. They can roughly be categorized as follows:

1. Work products describing the system design, which are
   - Requirements, including safety requirements and safety goals
   - System architecture (models)
2. Plans for safety activities, including verification plans and the safety plan itself
3. Work products regarding analysis, verification, and validation activities
   - Specifications for V&V activities, e.g., test vectors and tool configurations
   - Analysis results
   - Reports
4. The safety case
5. The product being developed (code, software, manuals, ...)
6. Work products concerning the process, e.g., change requests

All work products must be documented. ISO 26262 defines requirements on documentation of work products. The layout and medium of work products and documentation is not fixed, as long as the information required by ISO 26262 is contained in a clear, structured and comprehensive way. Each document needs at least a title, an ID, an author and approver, a change history, and a status [ISO11b].

5.1.4 Requirements Structure

ISO 26262, Part 8 [ISO11b] requires that safety requirements are hierarchically structured according to the design phases (i.e., safety goals at the top, and safety requirements for software units and hardware parts at the bottom). Within each level, the requirements shall be grouped according to the elements of the architecture they belong to. ISO 26262 recommends to use a requirements management tool. Requirements must be traceable

- to the next upper level requirements they are derived from
- the derived requirements on the next lower level
- verification activities for the requirements

Furthermore, each requirement must have the following attributes [ISO11b]

- a unique ID that is stable during the development process
- a status
- an ASIL

The range of the status attribute is not defined by ISO 26262.

5.1.5 Verification & Validation

V & V activities are an important part of the ISO 26262 safety livecycle. The V&V activities need to be planned beforehand, including specification of inputs (e.g., test data), configurations, and pass/fail criteria. The V&V results need to be documented with a reference to the specification and verified work products (e.g., requirements), and clear statements about the outcome of the activities, including follow-up actions [ISO11b]. Both the V&V plans and the V&V reports are the basis for the safety assessment at the end of the safety lifecycle.

5.1.6 Conclusion

ISO 26262 defines a safety lifecycle that shall be performed as part of the development process of automotive HW/SW systems. It imposes the following requirements:

- *Requirements on the development process:*
  - Use of a requirements management tool
  - Support for formal methods
  - Configuration and change management of all work products according to development standards
  - Planning of development activities before their execution, including V&V activities
  - Documentation of all work products and development activities, including planning of development and V&V
- *Requirements on models:*
  - Unique IDs for all elements; IDs shall be stable during the development process
  - ASIL attributes of the form $X\,(Y)$ (with $X, Y \in \{QM, A, B, C, D\}$) for architecture elements and requirements
  - Traceability
    - among requirements (refinement links)
    - between requirements and architecture
    - between requirements and V&V artifacts
  - Support for formal requirements

  - Specification of test data and configuration for V&V tools

- *Requirements on V&V tools:*
  - Repeatable tests and analyses
  - test/analysis reports with traceability to
    - elements that have been verified
    - other inputs
    - tool configuration
  - Clear pass/fail results

5.2 Collaboration Requirements

This section summarizes the results of the Collaboration Workshop in Stockholm (cf. section 2.2). The goal of the workshop was to understand the partners' requirements on future collaborative processes.

5.2.1 Understandability

As an engineer in a model-based, collaborative development environment, I want to ...

C-1.1 ... understand the models I am working with even if they are large and complex.

C-1.2 ... work on clear abstraction levels making sure I do not get overwhelmed by too much information.

C-1.3 ... have tools to navigate large and complex models efficiently.

C-1.4 ... have reports on analysis results in order to understand complex simulations easily.

C-1.5 ... be able to compare analysis results easily in order to make informed design decisions.

C-1.6 ... have a versioning system for the created models to be able to follow their history.

C-1.7 ... have a versioning system to be able to create a safety case.

C-1.8 ... have traceability between requirements, models, and other artifacts to be able to understand design decisions.

C-1.9 ... have traceability between models and other artifacts to be able to create a safety case.

C-1.10 ... document design decisions as part of the model.

C-1.11 ... have a common glossary in a collaborative development project to avoid misconceptions with other team members.

5.2.2 Information Exchange

As an engineer in a model-based, collaborative development environment, I want to ...

C-2.1 ... exchange requirements (functional, timing, safety, security, ...) to create a common understanding of the system under development and to identify the problems to be solved.

C-2.2 ... exchange system, software, and hardware architectures to streamline collaborative development.

C-2.3 ... exchange safety-related information (e.g., hazards, risks) to be able to create a safety case.

C-2.4 ... exchange interface specifications to facilitate collaborative system development.

C-2.5 ... exchange analysis results in order to reveal problems and make informed collaborative design decisions.

C-2.6 ... have a standardized exchange data format to enable a tool-independent exchange.

C-2.7 ... have a machine readable exchange data format to ease import and export procedures.

C-2.8 ... maintain exchanged information under version control.

C-2.9 ... have partial or full shared access storage to facilitate exchange.

Intellectual Property Protection

C-2.10 ... extract interface specifications of existing models as accurate as possible in order to enable an efficient protection of the intellectual property contained in the underlying functional models.

C-2.11 ... automatically remove components or information previously marked as confidential.

C-2.12 ... define certain views and abstraction levels that can be removed or hided in later development phases.

Common Knowledge

C-2.13 ... define specific areas of non-competition to enable an open exchange of common knowledge.

5.2.3 Integration into Design Processes

As an engineer in a model-based, collaborative development environment, I want to ...

Alignment of Development Processes

C-3.1 ... work in an environment with aligned development processes between the partners involved.

C-3.2 ... work in an environment with clearly defined team roles.

C-3.3 ... rely on best practices for selected development phases.

C-3.4 ... conserve design decisions to ease the integration of work products.

C-3.5 ... have regular meetings with representatives from all incorporated teams to exchange information about tasks or problems.

Integration of Received Information

C-3.6 ... work in standardized tool environments to ease the integration of exchanged information and to reduce the overall integration effort.

C-3.7 ... make use of simple and clear connections between requirement engineering and implementation tools in order to keep the overview.

C-3.8 ... provide dedicated information (e.g., built-in documentation, standard interfaces) to help others integrate my work results.

C-3.9 ... receive dedicated information (e.g., built-in documentation, standard interfaces) to help me integrate the work results of others.

C-3.10 ... use simple mechanisms for importing and exporting existing data and legacy models to ensure a seamless and smooth introduction.

Brownfield Integration

C-3.12 ... reuse existing models to save effort and reduce error proneness.

C-3.13 ... be aware of all relevant licenses and legal information to avoid problems in later design and implementation phases.

5.2.4 Living Models

As an engineer in a model-based, collaborative development environment, I want to ...

C-4.1 ... make use of a change impact analysis in order to be able to estimate the effects of individual changes.

C-4.2 ... integrate analysis results into models to keep the overview.

C-4.3 ... use the analysis results to further develop and improve the model in order to increase its quality in a more focused way.

C-4.4 ... receive automatic updates at specified time points in case of changes that affect system components that I am responsible for.

5.2.5 Compliance to Standards

As an engineer in a model-based, collaborative development environment, I want to ...

Standard-compliant Design Processes

C-5.1 ... follow a standard-compliant design process.

Development of Standard-compliant Systems

C-5.2 ... develop standard-compliant systems.

6 Gap Analysis between Requirements for Collaborative Systems Engineering, State-of-the-Art and State-of-Practice

After having described the state of the art and the state of practice in collaborative requirements engineering as well as the requirements practitioners have on such processes, we now proceed to analyse which gaps exist between these aspects. In section 6.1, we first outline the open issues in the ARAMIS II process w.r.t. the development of safety-critical systems according to ISO 26262. In section 6.2, we then tabulate the gaps based on the requirements we collected and a comparison with what we have described in Chapters 3 and 4.

6.1 Alignment of ISO 26262 with ARAMiS II Generic Process

As already described in section 5.1, the ISO 26262 standard specifies concrete requirements for both the *activities* to be performed and the *work products* to be provided. In Figure 6.1, an overview on the alignment of the safety process with the ARAMiS II generic process is given. These are the results of the gap analysis outlined in section 2.4. As described in section 2.4, because of the ARAMiS II methodology, the matching clauses are somehow supported by the AMALTHEA meta-model (except for the hazard analysis).

In addition, the ISO 26262 parts and clauses are highlighted that are covered by the design process developed in the AMALTHEA4public project [AMA17] and therein are also supported by the AMALTHEA meta-model, as presented in [TMSP16]. Taking into account the limitations described in section 2.4, it becomes clear that the two design processes relate to similar ISO 26262 parts, but that they complement each other well due to their focuses on different clauses. As figure 6.1 shows, both processes use the AMALTHEA meta-model largely in the same parts of the safety process, which are mainly system and software development. This also matches the results from the survey performed within the PANORAMA project presented in chapter 4, where the AMALTHEA meta-model is used in the concept and software development phases.

The use of AMALTHEA in the state-of-practice (chapter 4) seems to be quite limited to exchange of timing specifications and software architectures, whereas it has been used in AMALTHEA4public and ARAMiS II in a broader context. In AMALTHEA4public and ARAMiS II, AMALTHEA has been used also for specification of functional architecture, as well as some safety requirements [TMSP16; ARA19f]. In the context of the above gap analysis against ISO 26262, a closer look has been taken to the AMALTHEA elements and

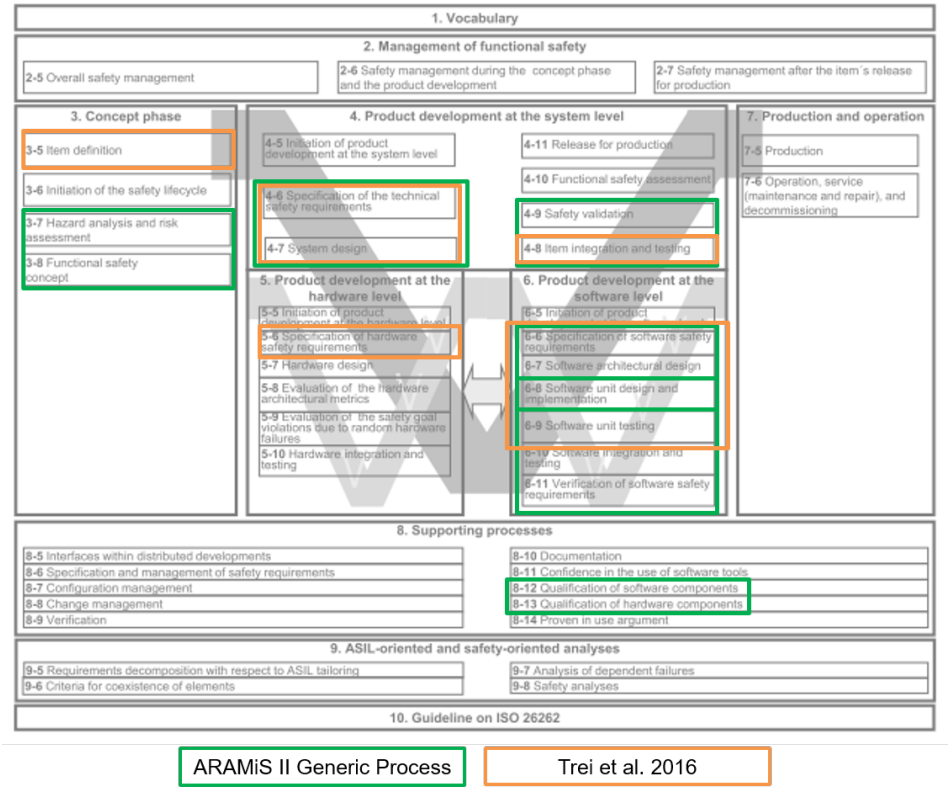

Figure 6.1: Alignment of ISO 26262 with ARAMiS II Generic Process

extensions used for specifying safety requirements. Comparing them to the requirements from ISO 26262 regarding layout and management of requirements shows that they lack a unique ID, ASIL and status information. It seems more appropriate to use a dedicated requirements management tool for managing safety requirements and linking them against AMALTHEA, instead of specifying them directly within the model. Using requirements management software is recommended by ISO 26262 and state of practice.

As a conclusion, evaluating state-of-the-art and state-of-practice development processes in context of ISO 26262 matches the surveys performed in PANORAMA (see chapter 4 and section 5.2) so far. It confirms the role of AMALTHEA as an exchange format within distributed development, but on the other hand highlights the need of tools that provide traceability between AMALTHEA and related design artifacts such as safety requirements.

6.2 Gaps Between Practitioner Requirements and State of the Art/State of Practice

In the following sub-sections, we present the analysis of the gap between the requirements collected from practitioners as listed in section 5.2 and the state of the art and state of the practice as discussed in chapter 3 and chapter 4. We follow the same structure that we used to group the requirements.

6.2.1 Understanding

Table 6.1: Gaps between requirements for understanding and state of the art/practice

Requirement	**State of the Art/Practice**	**Gap**
C-1.1 ... understand the models I am working with even if they are large and complex.	To understand large models, abstractions of the models are created. Due to missing tool support, this is done mostly manually. This has of course the increased risk to have inconsistencies in the abstract views.	• Tool support to create abstract views on large models automatically • Methodological and tool support to refine models
C-1.2 ... work on clear abstraction levels making sure I do not get over-whelmed by too much information.	see C-1.1	see C-1.1
C-1.3 ... have tools to navigate large and complex models efficiently.	In current practice, tool support for navigating large models is very limited. Therefore, graphical representation of the information represented by the models (e.g. diagrams or tables) are exported as pictures and stored in documents	• Tool support to created different views on large-scale models • Possibilities to work collaboratively on models and to merge and diff information
C-1.4 ... have reports on analysis results in order to understand complex simulations easily.	In currently industrial practice analysis results are stored in documents which are created manually.	Enable automated report generation based on the provided models
C-1.5 ... be able to compare analysis results easily in order to make informed design decisions.	Since analysis results are stored in documents, today the comparison of different analysis results must be performed mentally by experts which then make decisions.	• Store analysis results in form of a model • Provide tool support to link analysis results to requirements and check if the requirements are fulfilled • Capture design decisions explicitly in the model itself

Requirement	**State of the Art/Practice**	**Gap**
C-1.6 ... have a versioning system for the created models to be able to follow their history.	Current versioning systems used are Git and SVN which all work on text level. When models are shared between different companies there are cases where one repository is used to store the models and both OEMs and suppliers have access to the repository. However, in many cases, the same model is maintained in different repositories due to access reasons.	• There is a need for version control systems that support model versioning • Processes and workflows are needed to support management of models in different repositories
C-1.7 ... have a versioning system to be able to create a safety case.	In current practice, safety cases are created in form of documents (e.g. in MS word) and reference other specification and report documents. All of these documents are stored in document management systems (e.g. SAP).	• Create and maintain safety cases in form of models • Provide fine-granular links to models which represent the system
C-1.8 ... have traceability between requirements, models, and other artifacts to be able to understand design decisions.	Most common form of traceability using unique IDs as tags within the artifacts. In some traceability products, tools are used that allow links to be represented as hyperlinks that can be navigated.	• Missing traceability between artifacts of different types e.g., requirements and design • Lack of useful traceability visualisation. E.g., visualisation of relationships between faults, failures and system elements.
C-1.9 ... have traceability between models and other artifacts to be able to create a safety case.	Excel sheets are used to manage relationships between safety requirements and safety engineering processes (i.e FMEA, FTA)	• Create models to represent safety aspects of the system to allow for safety analysis • Export models with fault propagation links between software components
C-1.10 ... document design decisions as part of the model.	Design decisions are taken implicitly by experts and are rarely documented	Capture design decisions explicitly in the model itself
C-1.11 ... have a common glossary in a collaborative development project to avoid misconceptions with other team members.	Each team has its own set of terminology and therefore a common glossary does not exist	Support adding faults and/or failures, using a pre-defined glossary according to the type of amalthea element we are tracing with (i.e. specific glossary for software faults)

6.2.2 Information Exchange

Table 6.2: Gaps between requirements for information exchange and state of the art/practice

Requirement	State of the Art/Practice	Gap
C-2.1 ... exchange requirements (functional, timing, safety, security, ...) to create a common understanding of the system under development and to identify the problems to be solved.	Both functional and non-function safety requirements are exchanged in standard formats such as XML, documents, spreadsheets. Requirement management tools such as polarion helps in tracking and traceability of requirements.	• Validation check if requirements meet specific standards (ISO26262) • Porting of existing requirements and deriving work products specific to a ISO standard • Integrating safety requirements to the existing standard Amalthea exchange format (ASIL indicators etc)
C-2.2 ... exchange system, software, and hardware architectures to streamline collaborative development.	Amalthea model is used as standard exchange format to exchange system, software and hardware architectures in some collaborations (Tier 1 and Tier2, Tier 1 and OEM's). Proprietary formats supported by tool providers also capture system information and are also exchanged between partners. In some cases model transformations are performed to support the import of standard exchange formats in the proprietary tools. In some cases tools support direct import of standard formats.	
C-2.3 ... exchange safety-related information (e.g., hazards, risks) to be able to create a safety case.	Safety related information (hazards, requirements, results) are exchanged in the form of documents.	• Creation of correct safety cases for the overall system (after integration of sub-components) • Machine-readable format is required to formalize the safety information and to automate the exchange the process
C-2.4 ... exchange interface specifications to facilitate collaborative system development.	Standards such as AUTOSAR describe interfaces between different software components with a exchange format (arxml). This format is exchanged between OEM's, Tier1 and Tier2 suppliers.	

Requirement	State of the Art/Practice	Gap
C-2.5 ... exchange analysis results in order to reveal problems and make informed collaborative design decisions.	Requirement violations and analysis results are exchanged in the form of standard HTML files.	Capture analysis results in the model which is used as basis to perform the analysis
C-2.6 ... have a standardized exchange data format to enable a tool-independent exchange.	At various stages of system design and development different tool models are exchanged (e.g.,: MATLAB/ Simulink, SysML). Tool-independent exchange of models often not working correctly.	
C-2.7 ... have a machine readable exchange data format to ease import and export procedures.	Tool providers offer several such import and export features to support different formats (contradicting to C-2.6). Requirement looks more generic.	
C-2.8 ... maintain exchanged information under version control.	Requirements, models are version controlled (SVN, GIT)	Analysis results corresponding to each model are not version controlled.
C-2.9 ... have partial or full shared access storage to facilitate exchange.	Confluence, Shared GIT repository are used to facilitate exchange	

6.2.3 Intellectual Property Protection

Table 6.3: Gaps between requirements for intellectual property protection and state of the art/practice

Requirement	State of the Art/Practice	Gap
C-2.19 ... extract interface specifications of existing models as accurately as possible in order to enable an efficient protection of the intellectual property contained in the underlying functional models.	"In terms of information security, it is not uncommon that sensitive information needs to be redacted from an artifact before it is exchanged. This can include **information about the function of a model**, **the actual functionality and algorithmic details**, performance parameters, data accuracy, and sensor rates." (cf. chapter 4)	Requirement C-2.19 falls into the category of information hiding. The state of practice of accurate interface specification extraction was not mentioned explicitly though. From PANORAMA perspective, accurate interface extraction based on open source tools is not yet available.

Requirement	**State of the Art/Practice**	**Gap**
C-2.11 ... automatically remove components or information previously marked as confidential.	"[Information security] needs are usually captured in the initial contract and assured throughout by, e.g., creating different models of the same component with different level of detail, hiding information by replacing identifiers with placeholders, imposing strict access control on sensitive information, or encrypting information when exchanging it via insecure channels. In some cases, it is also possible to exchange binaries instead of source code or create special 'opaque' models that cannot be introspected." (cf. chapter 4)	From the elicited data it is unclear whether information hiding is done automatically in practice. From PANORAMA perspective, automatic information hiding based on open source tools is not yet available.
C-2.12 ... define certain views and abstraction levels that can be removed or hidden in later development phases.	"In terms of information security, **it is not uncommon that sensitive information needs to be redacted from an artifact before it is exchanged**. [...] These needs are usually captured in the initial contract and assured throughout by, e.g., creating different models of the same component with different level of detail, hiding information by replacing identifiers with placeholders, imposing strict access control on sensitive information, or encrypting information when exchanging it via insecure channels." (cf. chapter 4)	The state of practice elicitation shows that information hiding is actively used in order to protect sensitive information in models already. From PANORAMA perspective, dedicated abstraction levels or views to protect sensitive information are not yet available.

6.2.4 Common Knowledge

Table 6.4: Gaps between requirements for common knowledge and state of the art/practice

Requirement	State of the Art/Practice	Gap
C-2.13 ... define specific areas of non-competition to enable an open exchange of common knowledge.	The type of artifacts exchanged between different partners encompasses bi-directional model exchange in Amalthea format (cf. chapter 4). "If collaboration takes place within a supply chain, e.g., for a vehicle or an aircraft, there are different approaches to managing the contracts. Either the OEM sub-contracts development to a Tier-1 supplier, which in turn sub-contracts to Tier-2 suppliers, etc., or the OEM coordinates the collaboration with appropriate contracts all the way down the supply chain. While this depends on the type of the project, in practice it also depends on the existing relationship between the partners. The existing relationship also determines the level of formality of the contract which can range from a loose general agreement to a full-fledged formal contract between partners." (cf. chapter 4)	Although open formats are used to exchange information, the state of practice of Common Knowledge definition was not mentioned explicitly in the elicitated state of practice. The focus group results indicate a mostly bilateral information exchange that is constrained by the contracts between partners. We conclude that the definition of Common Knowledge in context of PANORAMA models is an open task.

6.2.5 Alignment of Development Process

Table 6.5: Gaps between requirements for alignment of development process and state of the art/practice

Requirement	State of the Art/Practice	Gap
C-3.1 ... work in an environment with aligned development processes between the partners involved.	Different partners in a development process (usually suppliers and OEMs) do have defined roles, defined by the contractual obligation and often build on a long-standing relationship. Interaction between the partners is common during development with differing degrees of intensity, in particular for mature partnerships. Synchronization between the processes happens (cf. C-3.3), but processes are usually not aligned in terms of sprint length, etc. (cf. chapter 4).	None.

Requirement	**State of the Art/Practice**	**Gap**
C-3.2 ... work in an environment with clearly defined team roles.	The roles in a collaborative work environment are usually clearly defined: "Roles that are typically involved in managing the collaborative effort are project managers and team leaders, technical managers, and business development managers. Depending on the project, engineers often have direct interaction with each other. In some cases, a dedicated engineer is co-located with the collaboration partners or directly responsible for handling a specific customer." (cf. chapter 4)	None.
C-3.3 ... rely on best practices for selected development phases.	Existing best practices are synchronisation points between the partners that help to align the processes and guidelines about when to increase and decrease collaboration intensity. The way artifacts are exchanged is also standardised to a degree (cf. C-2.X).	
C-3.4 ... conserve design decisions to ease the integration of work products.	It is not clear to which degree this is done at this point. In principle, such information can be recorded and exchanged based on an agreement between the parties (cf. C-3.8).	
C-3.5 ... have regular meetings with representatives from all incorporated teams to exchange information about tasks or problems.	This already seems to happen to a degree, in particular when conflicts arise or difficult technical issues need to be resolved.	

6.2.6 Integration of Received Information

Table 6.6: Gaps between requirements for integration of received information and state of the art/practice

Requirement	State of the Art/Practice	Gap
C-3.6 ... work in standardized tool environments to ease the integration of exchanged information and to reduce the overall integration effort.	Toolsets are tailored towards the specific tasks and often to the specific customer (cf. C-3.10). Individual tools are connected by "glue tools", often command-line tools, that convert formats and copy files to the correct positions. This makes the tool chains hard to set up and the development process error prone.	A standardized tool environment requires: • tools flexible enough to support different concrete projects; • standardised exchange formats for the most common artifacts • extensibility to address customer-specific requests. APP4MC strives to support these requirements at least for the purposes of timing modelling and analysis. Other tools such as Siemens Polarion have similar aims.
C-3.7 ... make use of simple and clear connections between requirement engineering and implementation tools in order to keep the overview.	The common way to establish connections between artifacts in different lifecycle phases is the use of identifiers. These identifiers can be mined to create requirements traceability matrices and other artifacts necessary for validation and certification and can be navigated if tool support exists. There is no explicit trace model and little support for change impact analysis or other activities. Traceability is not a part of the developers' daily work since trace links are not easily accessible.	The state of the practice does not yet allow to use trace information for program comprehension or to maintain an overview of the system. Work within Panorama on traceability management addresses this gap.
C-3.8 ... provide dedicated information (e.g., built-in documentation, standard interfaces) to help others integrate my work results.	At this point, which information is exchanged depends on the agreement between the involved parties. Each party can decide to publish this information and an agreement should be made about the format in which it is published.	Models should provide possibilities to capture the information on which the involved parties agree to exchange. If this are specific information only relevant for a dedicated information exchange a standardized (see C-2.6) must be extensible to represent the specific information.

Requirement	State of the Art/Practice	Gap
C-3.9 ... receive dedicated information (e.g., built-in documentation, standard interfaces) to help me integrate the work results of others.	See C-3.8.	See C-3.8.
C-3.10 ... use simple mechanisms for importing and exporting existing data and legacy models to ensure a seamless and smooth introduction.	In practice, many companies build specialised toolsets to process the data they get from their customers. The format of the exchanged data is not standardised and depends on who is involved in the process. While most tools offer some common exchange formats (e.g., ReqIF, EMF UML), they are not routinely used for data exchange, partially because they are not able to support the full semantics and partially because customers cannot provide the data in these formats. (cf. chapter 4)	There is no overarching standard for data exchange. Attempts like OSLC have not ultimately led to an improvement of the situation and the exchange of data in proprietary formats is still widespread. The Amalthea meta-model might be used as a defacto standard for some relevant information in the development process, but this overall problem is not in the scope of Panorama.

6.2.7 Brownfield Integration

Table 6.7: Gaps between requirements for brownfield integration and state of the art/practice

Requirement	State of the Art/Practice	Gap
C-3.12 ... reuse existing models to save effort and reduce error proneness.	Today, reuse of models is not always possible due to limited tool support and clone-and-own approaches a primarily used.	• Tools must support the storing of model elements in a repository • Reuse concepts must be explicitly defined and implemented in the tool (e.g. black-box vs. white-box)
C-3.13 ...be aware of all relevant licenses and legal information to avoid problems in later design and implementation phases.		

6.2.8 Living Models

Table 6.8: Gaps between requirements for living models and state of the art/practice

Requirement	State of the Art/Practice	Gap
C-4.1 ... make use of a change impact analysis in order to be able to estimate the effects of individual changes.	Traceability links are mostly maintained using unique IDs of the different artifacts. This makes impact analysis hard since artifacts have to be investigated manually (cf. C-3.7).	• Integration/aggregation of the traceability links so that they can be used for impact analysis • Flexible visualisation of traceability links in order to facilitate traceability links
C-4.2 ... integrate analysis results into models to keep the overview.	Analysis results are a separate report that need to be analysed in order to identify what needs to be improved in the model	Tools that integrate analysis results into the model
C-4.3 ... use the analysis results to further develop and improve the model in order to increase its quality in a more focused way.	Analysis results are a separate report that need to be analysed in order to identify what needs to be improved in the model	• There is a need to link analysis results to quality attributes that can be improved in the model • There is a need to develop tool support that provide recommendations on what can be improved in the model based on current analysis results
C-4.4 ... receive automatic updates at specified time points in case of changes that affect system components that I am responsible for.	• Changes need to be identified manually when requirements change. For changes that have an impact on the monetary value of the contract, more formal discussions are needed before changes can be made. • In cases where simulation models are used, the models can be used to explore how the requirements changes will affect the current model • Tools like Polarion support change management and send notifications to stakeholders when artifacts change	While there are tools that support change management and provide notification mechanisms, it is rare that a common tool is used throughout the company or between the different companies that need to collaborate. Mechanisms to support change management that go beyond a specific tool are therefore required.

6.2.9 Standard-compliant Design Process

Table 6.9: Gaps between requirements for standard-compliant design process and state of the art/practice

Requirement	State of the Art/Practice	Gap
C-5.1 ... follow a standard-compliant design process.	There is no standard collaborative design process in place at the moment. The process is defined and adapted by the suppliers for each customer. The process therefore ranges from agile and iterative to plan driven processes like waterfall and in some cases hybrid processes are used.	The need to adapt a process for each customer can be problematic for the supplier, since they cannot follow one process. A suggested solution is to have a standard process in-house and have a way to synch this standard process with each supplier. However, the feasibility of this needs to be investigated.

6.2.10 Development of Standard-compliant Systems

Table 6.10: Gaps between requirements for development of standard-compliant systems and state of the art/practice

Requirement	State of the Art/Practice	Gap
C-5.2 ... develop standard-compliant systems.	All safety critical products need to adhere to safety standards. The automotive partners follow ISO 26262 while the aviation partners follow DO 178B. This means that for such systems, extra artifacts e.g., extra documentation or traceability need to be defined. This also affects the process that needs to be followed since it has to comply to the safety standards. More effort is put towards verification and validation.	More tool support is required by practitioners to help them define safety assurance cases and perform safety analysis. Practitioners need the ability to formally model safety requirements (c.f C-2.3) in order to automate safety analysis.

7 Summary and Conclusion

Our analysis shows that a number of gaps persist between the requirements practitioners express for collaborative systems engineering processes and what the state of the art and the state of practice has to offer. However, the state of the art describes a number of approaches to address such gaps. What remains are mostly practical issues that revolve around incompatible tool chains and a lack of exchange formats. In situations in which partners exchange and update information frequently, the seamless exchange of data and integration into the development, visualisation, and test environments is crucial.

At the same time, such free exchange of information also introduces new problems in terms of privacy and security of data, protection of intellectual properties, and licensing. Furthermore, the need to collaboratively define and maintain safety cases means that partners need to apply rigour and need to be able to track changes. Updated information from partners also needs to be integrated into the traceability models to show that all requirements are tested and to enable change impact analysis.

The solutions that PANORAMA can provide in this space are three-fold:

Exchange format: PANORAMA is going to provide the Amalthea model, an exchange format that might become the defacto standard for exchange of timing information for heterogeneous embedded systems. While this is only one of several aspects that requires an exchange format, the project focus lies here. By providing transformations from and to the Amalthea model, it will also be possible to derive data from other languages, such as AUTOSAR and EAST-ADL and to interface with a variety of other tools.

Process definition: A combination of process knowledge from Amalthea4public, DEIS, and ARAMIS II will provide the foundation for the project going forward. In particular, the existing use cases, the process descriptions, and tools such as the Open Dependability Exchange meta-model [DEI19] will be reused and refined for the purposes of PANORAMA.

Security: PANORAMA will deliver new results in terms of protecting private data and specific intellectual properties in collaborative development processes. A first step will be a methodology for the analysis of data privacy threats in such settings.

Safety: Another project focus is the collaborative, model-driven creation and maintenance of safety cases. For this purpose, first steps are being made to identify the models required and which analysis steps need to be executed in the context of a generic process definition for collaborative systems engineering.

Traceability: In terms of traceability, we will deliver results for decision support to create and maintain traceability links. We will tailor this particularly to situations in which traced artifacts are replaced during development and the existing trace links to these artifacts need to be reestablished.

The main contribution of this deliverable, the gap analysis, will guide the development of these solutions. We also consider it as an initial analysis which will be completed and extended as the project progresses.

Bibliography

[AMA17] AMALTHEA4public Project Consortium, *AMALTHEA4public. An Open Platform Project for Embedded Multicore Systems*, `http://www.amalthea-project.org/`, 2017.

[AP10] A. Avritzer and D. J. Paulish, "A comparison of commonly used processes for multi-site software development," in *Collaborative Software Engineering*, Springer, 2010, pp. 285–302.

[Apa19] Apache Software Foundation, *Subversion*, `https://subversion.apache.org/`, 2019.

[ARA19a] ARAMiS II Project Consortium, *ARAMiS II. Development processes. Tools. Plattforms*, `https://www.aramis2.com/`, 2019.

[ARA19b] ——, "Generic multicore development process," Achievement E2.2, 2019, not yet published.

[ARA19c] ——, "Platform architecture characterization and mitigation / technical architecture models," Achievement E2.3, 2019, not yet published.

[ARA19d] ——, "Tool interoperability specification and methodology," Achievement E2.4, 2019, not yet published.

[ARA19e] ——, "Partitioning of software components," Achievement E3.3, 2019, not yet published.

[ARA19f] ——, "Avionics demonstrator II," Achievement E5.5, 2019, not yet published.

[BRH+13] J. Birch, R. Rivett, I. Habli, B. Bradshaw, J. Botham, D. Higham, P. Jesty, H. Monkhouse, and R. Palin, "Safety cases and their role in iso 26262 functional safety assessment," in *International Conference on Computer Safety, Reliability, and Security*, Springer, 2013, pp. 154–165.

[CrE19] CrESt (Collaborative Embedded Systems), *Concepts of CrESt Tools and Interoperability Framework*, `https://crest.in.tum.de/index.html`, 2019.

[DEI19] DEIS Project Consortium, "Digital dependability identities and the open dependability exchange meta-model," Deliverable D3.1, 2019. [Online]. Available: `http://www.deis-project.eu/fileadmin/user_upload/DEIS_D3.1_Specification_of_the_ODE_metamodel_and_documentation_of_the_fundamental_concept_of_DDI_PU.pdf`.

[FRMM18] M. Franzago, D. D. Ruscio, I. Malavolta, and H. Muccini, "Collaborative model-driven software engineering: A classification framework and a research map," *IEEE Transactions on Software Engineering*, vol. 44, no. 12, pp. 1146–1175, Dec. 2018, ISSN: 2326-3881. DOI: `10.1109/TSE.2017.2755039`.

[FVG19] A. Frigerio, B. Vermeulen, and K. Goossens, "Component-level asil decomposition for automotive architectures," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, IEEE, 2019, pp. 62–69.

[Git19] Git Community, *Git*, `https://git-scm.com/`, 2019.

[HGK+08] T. Hildenbrand, M. Geisser, T. Kude, D. Bruch, and T. Acker, "Agile methodologies for distributed collaborative development of enterprise applications," in *2008 International Conference on Complex, Intelligent and Software Intensive Systems*, IEEE, 2008, pp. 540–545.

[ISO11a] ISO/IEC 26262-5:2011, "Road vehicles—functional safety—part 5: Product development at the hardware level," International Organization for Standardization, Standard, 2011.

[ISO11b] ISO/IEC 26262-8:2011, "Road vehicles—functional safety—part 8: Supporting processes," International Organization for Standardization, Standard, 2011.

[ISO11c] ISO/IEC 26262-9:2011, "Road vehicles—functional safety—part 9: Automotive safety integrity level (ASIL)-oriented and safety-oriented analyses," International Organization for Standardization, Standard, 2011.

[ISO11d] ISO/IEC 26262:2011, "Road vehicles—functional safety," International Organization for Standardization, Standard, 2011.

[KS19] S. Kunz and T. Sandmann, *Modellbasierte Multicore-Softwareentwicklung. Die ARAMiS II Entwicklungsprozesse*, `https://www.aramis2.org/app/download/9075850976/04_ARAMiS_II_TP0_2019-09-20_Abschlussveranstaltung_TP2.pdf`, Sep. 2019.

[Roh69] B. Rohrbach, *Kreativ nach Regeln – Methode 635, eine neue Technik zum Lösen von Problemen*, Oct. 1969. [Online]. Available: `https://en.wikipedia.org/wiki/6-3-5_Brainwriting`.

[Siea] Siemens, *Polarion Extensions*. [Online]. Available: `https://extensions.polarion.com/?__hstc=2015854.53dfcc7898a1ee5a006c6faebd7303e8.1570789774181.1570789774181.1570789774181.1&__hssc=2015854.1.1570789774182&__hsfp=391108424`.

[Sieb] ——, *Polarion Platform*. [Online]. Available: `https://polarion.plm.automation.siemens.com/products/overview`.

[Sie16a] ——, *Accelerating automotive innovation*, White Paper, 2016. [Online]. Available: `https://polarion.plm.automation.siemens.com/resources/download/accelerating-automotive-innovation`.

[Sie16b] ——, *Automate lifecycle governance and compliance for automotive software systems*, White Paper, 2016. [Online]. Available: `https://polarion.plm.automation.siemens.com/resources/download/automate-lifecycle-governance-and-compliance-for-automotive-software-systems`.

[TMSP16] M. Trei, S. Maro, J.-P. Steghöfer, and T. Peikenkamp, "An ISO 26262 Compliant Design Flow and Tool for Automotive Multicore Systems," in *International Conference on Product-Focused Software Process Improvement*, Springer, 2016, pp. 163–180.

[WC12] D. D. Ward and S. E. Crozier, "The uses and abuses of asil decomposition in iso 26262," in *7th IET International Conference on System Safety, incorporating the Cyber Security Conference 2012*, IET, 2012, pp. 1–6.