ECSEL Joint Undertaking

European Commission

ECSEL Research and Innovation actions (RIA)

AMASS

Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems

Baseline and requirements for cross/intra-domain reuse
D6.1

Work Package:	WP6: Cross/Intra-Domain Reuse
Dissemination level:	PU = Public
Status:	Final
Date:	9 March 2018
Responsible partner:	Barbara Gallina (Mälardalen University)
Contact information:	barbara.gallina@mdh.se
Document reference:	AMASS_D6.1_WP6_MDH_V1.1

This deliverable is part of a project that has received funding from the ECSEL JU under grant agreement No 692474. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and from Spain, Czech Republic, Germany, Sweden, Italy, United Kingdom and France.

Contributors

Names	Organisation
Barbara Gallina, Irfan Sljivo, and Inmaculada Ayala	Mälardalen University (MDH)
Anna Carlsson	OHB Sweden (OHB)
Jose María Álvarez, Jose Luis de la Vara, Elena Gallego, and Pablo Sánchez	Universidad Carlos III de Madrid (UC3)
Jose Miguel Fuentes and Borja López	The Reuse Company (TRC)
Helmut Martin and Bernhard Winkler	Virtual Vehicle (VIF)
Huascar Espinoza and Alejandra Ruiz	TECNALIA Research & Innovation (TEC)
Ines Hoffman and Michael Soden	KPIT medini Technologies AG (KMT)
Ramiro Demasi and Stefano Tonetta	Fondazione Bruno Kessler (FBK)
Andrea Musone	Intecs (INT)
Staffan Skogby	ALTEN (ALT)
Andreas Preussger, Frank Badstuebner, and P. Stirgwolt	Infineon (IFX)

Reviewers

Names	Organisation
[Peer Reviewer] Silvia Mazzini	Intecs (INT)
[Peer Reviewer] Erwin Schoitsch	AIT Austrian Institute of Technology GmbH (AIT)
Jose Luis de la Vara	Universidad Carlos III de Madrid (UC3)
Petr Böhm	AIT Austrian Institute of Technology GmbH (AIT)

Document History

Version	Date	Status	Author (Partner)	Remarks
V1.0	2016-09-26	Final	B. Gallina (MDH)	Submitted version
V1.1	2018-03-09	Final	B. Gallina (MDH)	Revised to consider comments from EC reviewers (June 2017).

TABLE OF CONTENTS

List of Figures

List of Tables

Executive Summary

This deliverable (D6.1 -Baseline and requirements for cross/intra-domain reuse) sets the stage of WP6. The stage is set by first of all recalling the AMASS context, motivation, and objectives. Then, the problem in its multifaceted nature is stated. In particular, a taxonomy related to the concept of reuse is used to illustrate the AMASS focus, which embraces process-, product-, and assurance case-related reuse. Then, via the analysis of both the state of the art and the state of the practice concerning intra and cross-domain reuse, relevant building block of the solution space are identified. More specifically, the analysis aims at allowing AMASS to adopt the best features from existing approaches and to guarantee compatibility. Moreover, when relevant, the state of the art and the state of the practice are compared in order to identify possible gaps. This comparative work ensures the identification of concrete needs, calling for new solutions, and ensuring the innovation of the project and future feasibility of exploitation of results.

Finally, a way forward is proposed. The proposed way forward consists of a consolidation of the existing results achieved within OPENCOSS, SafeCer and other ongoing and past projects, and of the available technology on the market and state of practice. More specifically, based on the three dimensions (process, product, assurance case) methods and technologies are identified to enable the systematization of reuse.
Concerning processes, EPF Composer-based solutions seem to be promising. Concerning products, contract-based reasoning, patterns, model-based principles applied to engineering, and variability management seem to be valuable approaches. Finally, concerning assurance cases, valuable approaches are: patterns, variability management, contract-based, and module-based argumentation approaches as well as model-based argumentation.

The stage, as set in this document, is going to be used during the execution of Task-6.2 and Task-6.3 of WP6. The AMASS deliverables D6.2 (Design of the AMASS tools and methods for cross/intra-domain reuse (a)) and D6.3 (Design of the AMASS tools and methods for cross/intra-domain reuse (b)), which are the expected outputs of Task 6.2 (Conceptual Approach for Cross-Domain and Intra-Domain Reuse), are expected to develop the way forward as envisioned within this document. This document is also going to be used during the execution of Task-3.2, Task-4.2, and Task-5.2, whenever reuse concerns have to be considered within respectively (sub)system specification, assurance case specification, and evidence management.

1. Introduction

Embedded systems have significantly increased in number, technical complexity, and sophistication, moving towards open, interconnected, networked systems (such as "the connected car" and the cloud), integrating the physical and digital world, thus justifying the term “cyber-physical systems” (CPS). This “cyber-physical” dimension is exacerbating the problem of ensuring safety, security, availability, robustness and reliability in the presence of human, environmental and technological risks. Furthermore, the products into which these Cyber-Physical Systems (CPS) are integrated (e.g. aircrafts) need to respect applicable standards for assurance and in some areas, they even need certification. The dimension of the certification issue becomes clear if we look at the passenger plane B 787 as a recent example – it is reported in [34] that the certification process lasted 8 years and has consumed 200.000 staff hours at the FAA, just for technical work. The staff hours of the manufacturer even exceeded this figure as more than 1500 regulations had to be fulfilled, with evidence reflected onto 4000+ documents. Although aircrafts are an extremely safety-critical product with many of such regulations, the situation in other areas (railway, automotive, medical devices etc.) is similar.

Unlike practices in electrical and mechanical equipment engineering, CPS do not have a set of standardized and harmonized practices for assurance and certification that ensure safe, secure and reliable operation with typical software and hardware architectures. As a result, the CPS community often finds it difficult to apply existing certification guidance. Ultimately, **the pace of assurance and certification will be determined by the ability of both industry and certification/assessment authorities to overcome technical, regulatory, and operational challenges. A key regulatory-related challenge has to be faced when trying to reuse CPS products from one application domain in another because they are constrained by different standards and the full assurance and certification process must be applied as if it were a totally new product, thus reducing the return on investment of such reuse decisions. Similarly, reuse is hindered often even within the same domain, when trying to reuse CPS products from one project to another, where assumptions change together with the criticality level.**

To face all these challenges, the AMASS approach focuses on the development and consolidation of an open and holistic assurance and certification framework for CPS, which constitutes the evolution of the OPENCOSS and SafeCer approaches towards an architecture-driven, multi-concern assurance, and seamlessly interoperable tool platform. The AMASS tangible expected results are:

a) The **AMASS Reference Tool Architecture,** which will **extend the OPENCOSS and SafeCer conceptual, modelling and methodological frameworks** for architecture-driven and multi-concern assurance, as well as **for further cross-domain and intra-domain reuse capabilities** and seamless interoperability mechanisms (based on OSLC specifications).

b) The **AMASS Open Tool Platform**, which will correspond to a collaborative tool environment supporting CPS assurance and certification. **This platform represents a concrete implementation of the AMASS Reference Tool Architecture, with a capability for evolution and adaptation, which will be released as an open technological solution by the AMASS project. AMASS openness is based on both standard OSLC APIs with external tools (e.g. engineering tools including V&V tools) and on open-source release of the AMASS building blocks.**

c) The **Open AMASS Community**, which will **manage the project outcomes, for maintenance, evolution and industrialization**. The Open Community will be supported by a governance board, and by rules, policies, and quality models. This includes support for **AMASS base tools** (tool infrastructure for database and access management, among others) and **extension tools** (enriching AMASS functionality). As Eclipse Foundation is part of the AMASS consortium, the **Polarsys/Eclipse community** (https://www.polarsys.org) is a strong candidate to host AMASS.

To achieve the AMASS results, as depicted in Figure 1, the multiple challenges and corresponding project scientific and technical objectives are addressed by different work-packages.

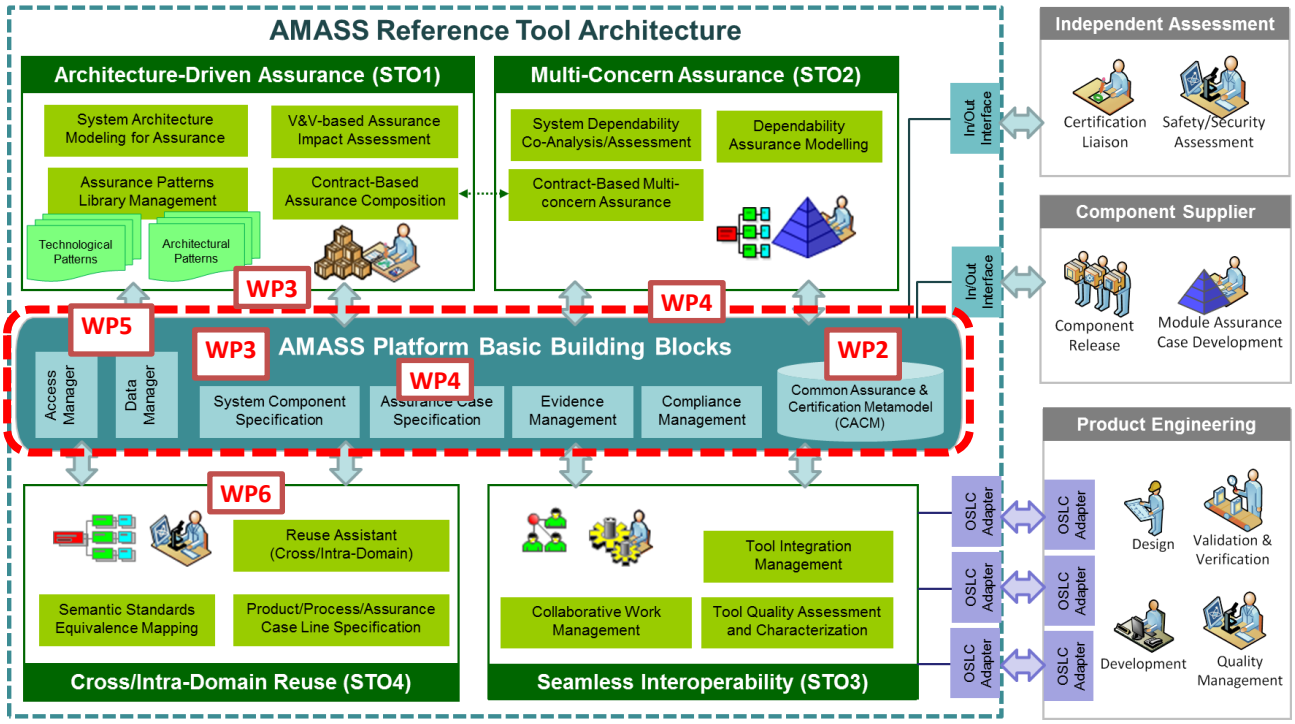

Figure 1: AMASS Building blocks

WP6 aims at addressing cross and intra-domain reuse. More specifically, with respect to the AMASS goals, this deliverable presents the background in terms of problem and solution space related to: Goal 2 (G2), the corresponding project objective O3, and to the project scientific and technical objective (STO) 4. G2, O3 and ST4 are recalled here to make the deliverable self-contained.

G2 demonstrates a potential reuse of assurance results (qualified or certified before), leading to 40% of cost reductions for component/product (re)certification/qualification activities.

O3 consolidates a cross-domain and intra-domain assurance reuse approach to improve mutual recognition agreement of compliance approvals and to help assessing the return of investment of reuse decisions.

STO4, which focuses on Cross/Intra-Domain Reuse, is constituted of three sub-objectives:

- **Semantics -based Standards Equivalence Mapping**: One obstacle to the cost-effective reuse of cross-domain assets is the fact that the terminology and semantics used to describe and manage assurance across different application domains are not consistent. For example, concepts such as ‘fault’, ‘hazard’ and ‘mishap’, and what constitutes a ‘component’ or a ‘subsystem’, overlap to some extent, but there are also differences between the definitions of these concepts across standards. OPENCOSS started to solve this issue by using the Common Certification Language (CCL) Vocabulary approach. The CCL Vocabulary is a structured and harmonized way to store and communicate knowledge about assurance artefacts and concerns. However, only exemplary cases have been explored with this approach. Within SafeCer, instead, an ontology-based method for process elements reuse has been explored [32]. AMASS will extend the OPENCOSS CCL Vocabulary through the AMASS Common Assurance and Certification Meta-model (CACM) by automating its creation and usage via a deepened application of the SafeCer ontology-based method. An automated CCL Vocabulary approach will also allow performing an informed gap analysis on the

standards and thus mitigate the risk of inappropriate reuse, when a given assurance asset does not appropriately match the requirements of the reuse context.

- **Reuse Assistant (Cross/Intra Domain**): In addition to the semantic mappings, we need to understand how the concepts work in terms of their relationship with one another to define the objectives of the standards, i.e. the intent of the requirements and process activities, and the artefacts they result in. In order to come to a clearer understanding of the role played by each activity and artefact in the overall assurance effort, AMASS will support users to understand whether reuse of the assurance assets is reasonable or determine what further analysis is required to justify claims of compliance. For example, AMASS will provide tool assistance to highlight the reasons why fault analysis is performed and the step in the development of the system at which it is applied (and hence the level of detail involved). The compositional argument approach developed by SafeCer and OPENCOSS will evolve to achieve the capability to characterize pre-existing argument modules in terms of the intent of the applicable standards. Clearly, this characterization will rely on a clear understanding and statement of the assurance objectives of each standard, and of the assurance assets used to evince the claims made to demonstrate their satisfaction.
- **Product/Process/Assurance Case Line Specification**: Variability management represents a problem for industry. Various methods have been developed to manage variability and thus to support industry in solving, or at least tackling, such a problem. No systematic approach is however available yet. A systematic approach is needed to deal with software/hardware variability management, but also with process and assurance case-related variability. The AMASS project will focus on extending and integrating the current methods in order to manage, for instance, the ripple effects on processes as well as assurance cases, as results of changes in product requirements. The objective is to promote a fully integrated approach addressing the fundamental dimensions for certification purposes.

In addition, WP6 is responsible for consolidating the previous works on compliance of standards in order to design and implement the basic building block called “Compliance Management” (Figure 1). Compliance management of safety-critical systems builds upon the industrial standards. That is: (a) what the standards define and (b) how compliance/conformance with the standards can be demonstrated.

To achieve STO4, WP6 is structured into three tasks. The purpose of this deliverable is to document the work conducted during Task 1 (Consolidation of the Current Approaches for Cross-Domain and Intra-Domain Reuse). More specifically, the purpose of the deliverable is multi-fold:

1) To analyse the problem related to reuse in order to understand its multifaceted nature;
2) To present the corresponding state of the art;
3) To present the current state of practice; and finally, based on these findings
4) To present a consolidation of existing results and profit from ongoing and past projects as well as available technology in the market are proposed.

Based on the investigated state of the art and state of practice approaches their gaps are identified to come up with a way forward enabling the formulation of requirements to achieve the reuse-oriented vision of AMASS covering multi-assurance, and seamless interoperability. This activity will serve to ensure both the innovation of the project and future feasibility of exploitation of results.

The rest of the deliverable is organised as follows. In Chapter 2, the problem concerning reuse is stated and its multifaceted nature is revealed. In Chapter 3, the state of the art on cross/intra domain reuse is presented. In Chapter 4, the state of the practice on cross/intra domain reuse is presented. Finally, in Chapter 5, a consolidation and the way forward in AMASS is envisaged.

2. Problem statement

The higher complexity and size of CPS products combined with the growing market demand requires industry to redefine its core and non-core activities, and to implement a coherent and systematic reuse strategy instead of relying exclusively on in-house developed and ad-hoc solutions. For example, if an engine control computer from the automotive industry has to be reused in the aerospace industry, as of today the full certification process is applied as if it were a completely new product, thus reducing the return on investment of such reuse decision. In such circumstances, systematic cross-domain reuse would be crucial to avoid the (redundant) application of the full certification process. In circumstances where a new version of a product comes from a previously certified version of that same product, systematic intra-domain reuse would also be crucial. Another aspect in which systematic intra-domain reuse would be crucial is in case of incremental certification (i.e. from a generic product to a specific one, obtained via addition of functionalities, or via utilization of configuration/calibration parameters).

2.1 Reuse Scenarios

We are facing a challenge related to the management of reuse, to the scope of reuse, and to the reuse methodology to be applied. For a better clarification of the problem, consider Figure 2.

Figure 2 illustrates a reuse taxonomy created to support the reader. The taxonomy is based on previous works by Ruiz in [85].

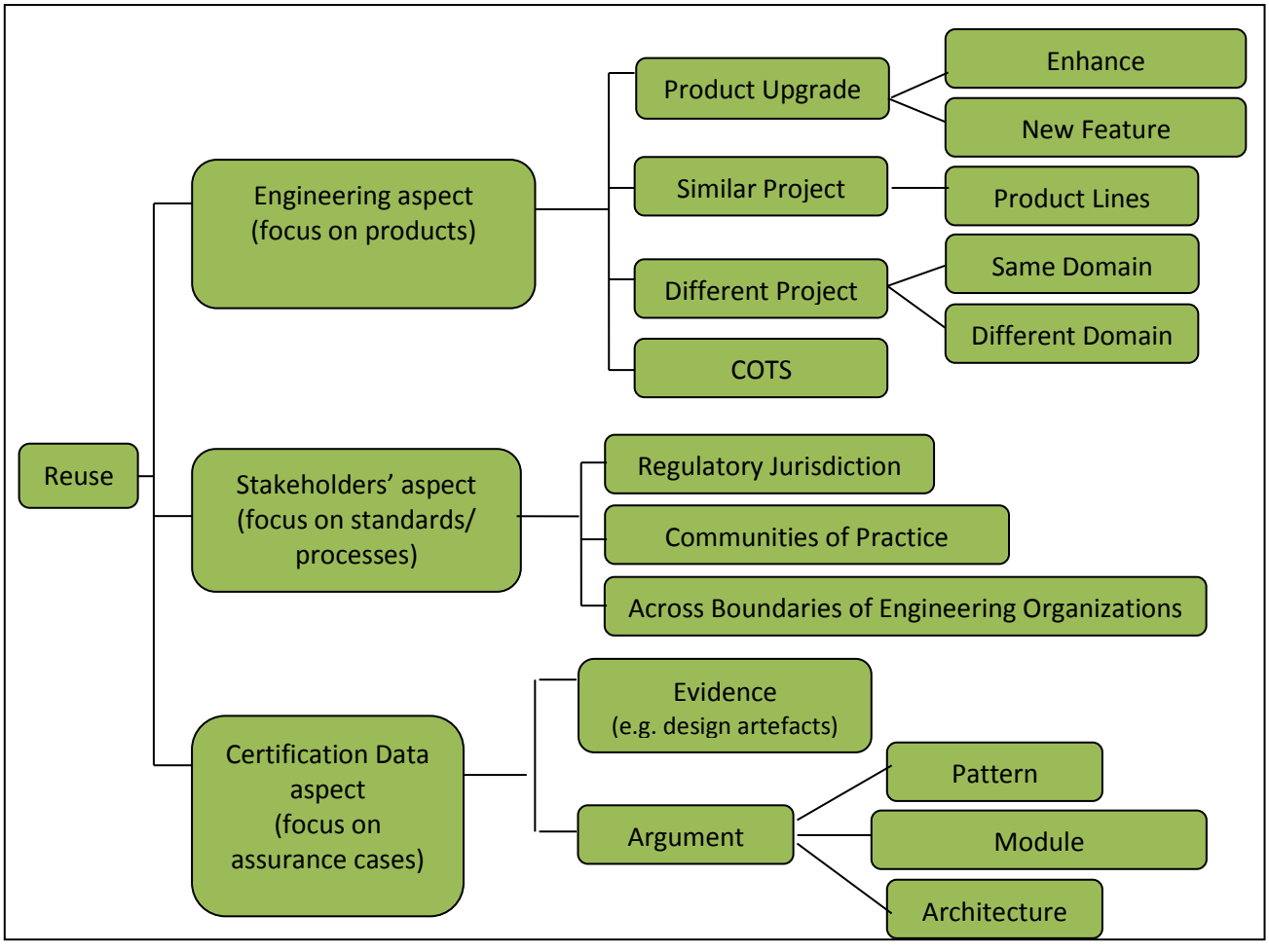

Figure 2: Reuse Taxonomy in the scope of AMASS

Each of the branches in the diagram shows a different type, or aspect, of reuse. There are scenarios that combine different reuse types, or aspects, depending on the perspective.

2.1.1 Engineering aspects

This aspect focuses on the idea of reuse of the engineering practices. This facet is based on the idea of engineering reuse, where the reuse is at technical level (design and implementation). In this situation the safety, security and related aspects or mechanism associated with the engineering decisions are reused. However, the reuse of work products of the concerned component (HW and/or SW), for demonstrating compliance with the applicable standards, is not straightforward, and some re-work is quite common. The possible reuse scenarios are the following:

- **(Same project) Upgrade – new feature**: A component associated with a particular hardware and/or software will include new features that the previous component did not have. We have a basic component that will include a new feature in the next version.
- **(Same project) Upgrade – Enhance performance**: A component associated with a particular hardware and/or software will be modified as part of its maintenance and will keep the same functionalities as the previous version but with enhanced performance.
- **Similar project**: A component is reused and integrated into a new system with the same context and domain, as previously used. The functionalities needed in both projects are the same and so the component is reused.
- **Similar Project - Product Lines**: A software product line [123] is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and are developed from a common set of assets in a prescribed way.
- **Different project - same domain**: A component developed for a specific project in a certain domain is reused in another project with a different context but both projects belong to the same domain. The operational environments and/or systems in which the component is integrated might be different.

  Note that the different contexts in which a differently configured component is deployed could to some extent be addressed via product line techniques. The set of differently configured components may also be interpreted as a product line and thus product line best practices can be used.
- **Different project - different domain**: General use components may be reused not only in projects from the same domain but also in projects from different domains.
- **(C)OTS:** In this case, the reuse is of a component described as "(Commercial) Off The Shelf". COTS components are usually general purpose ones that can apply to different domains and purposes. OTS are instead developed in house and may or may not cross the original domain. In the scientific literature, COTS are identified as Software of Unknown Pedigree (SOUP) [119] when their development has not followed the best practices mandated by the domain standards where they apply for being reused.

  Note that a COTS could be addressed via product line best practices since a COTS could to some extent be seen as a set of components with different configuration parameters.

2.1.2 Stakeholder aspect

This aspect focuses on the different interests/views depending on the involved stakeholders. The objective is not so much guided by the engineering decisions but on the standard compliance requisites and the best practices and methodology for a systematic reuse.

- **Regulatory jurisdiction**: Critical systems may operate in places where different jurisdictions apply, for example a plane landing in different countries. In this case, different jurisdictions apply to the same product/component and the certification artefacts generated for one jurisdiction can be used in order to achieve compliance with other jurisdiction(s). When components are expected to be

used in different countries, the different jurisdiction of each country shall be taken into account during the component design.

- **Communities of practice**: Reuse of the methodologies and practices related to one or more activities mentioned in a standard and shared between different communities with the same objectives
- **Across boundaries of engineering organizations**: Different groups within the same organization, or across different organizations, reuse the components (designs, implementation and/or certification artefacts) from one group to another.

The reuse between communities of practice and across boundaries of engineering organizations is in the scope of AMASS.

2.1.3 Certification data aspect

This aspect focuses on the reuse of certification related data (e.g., assurance cases) and work products (e.g., a design specification). The objective here is to reuse the data generated in previous projects or phases and minimise the repetition of certification related activities.

- **Evidence**: Reuse of the results from conducting an assurance (e.g., safety) related activity, procedure or applying a certain method or tool. The evidence of having conducted such activities is sufficient and there is no need to repeat them when reusing the component.
  Evidence can be classified into immediate, direct, and indirect.

  Note that Immediate evidence is defined as evidence that is itself being evaluated; the parts of the candidate artifact: machine executables, source code, specifications, requirements documents.

  Direct evidence is defined as evidence which presents properties of the candidate; things that are directly about the immediate evidence: test results, proofs of correctness, static analysis results, hazards analyses, real-world trials, model checking results.

  Indirect evidence is defined as evidence which describes the circumstances relevant to the creation of the candidate; information about the development of the artifact: development processes, personnel qualifications, tool qualifications, content management systems.

  Note also that Immediate and direct evidence represent artefacts that are produced during the development process and constituted the product [92]. Indirect evidence is an artefact produced to describe the development process and it is needed to show compliance [92].
    - **An example of immediate evidence are Design-related artefacts, i.e., an architectural specification**.
    - **An example of direct evidence are Verfication**-related artefacts, e.g.., model-checking results.
    - **An example of indirect evidence is a process** model.
- **Argumentation – Patterns**: Assurance case[1] patterns are considered as one of the main approaches for managing reuse of assurance. An assurance case pattern provides a means of explicitly and clearly documenting the structure of common reasoning as found in assurance cases, and it promotes the reuse of best practices for assurance.
- **Argumentation – Modules**: Assurance case's modules are parts of an overall assurance case containing part of an argument and relevant citations of evidence.

[1] Assurance cases and safety cases appear sometimes indistinctly in the literature. Safety cases are defined as "A structured argument, supported by a body of evidence that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given environment" [114]. However, these cases are not just limited to the safety area; we can find the previous concept applicable to IT trustworthiness [115], security [116] or compliance, conceiving assurance cases with a much broader scope. Assurance Case is defined as "a collection of auditable claims, arguments, and evidence created to support the contention that a defined system/service will satisfy the particular requirements" [117].

For instance, when contributing to an argument aimed at showing process compliance, an assurance case module may correspond to an interrelated set of assurance activities, scope of responsibilities of a particular engineering organisation, or well-defined sub-system or equipment used within the overall CPS platform.

- **Argumentation Architecture:** Safety case architecture is defined by Bates et al [124] as "the high organisation of the safety case into components of arguments and evidence, the externally visible properties of these components, and the interdependencies that exist between them". This definition can be generalized to assurance cases. Argumentation architecture supports the reuse of argument modules, which can be associated to both system and process components.

  Remark: Any other piece of argumentation that cannot be clearly identified as a module or pattern-instance, can generically be referred to as "fragment". A mandatory/alternative/optional branch within a family of assurance cases, for instance, could be labelled as fragment. The generation of a piece of argumentation that does not comply to any pattern and that is not encapsulated within any module can be labelled as "fragment".

2.2 Reuse Dimensions

All the above-listed reuse types are in the scope of AMASS. The above classification presents overlaps, which have been explicitly indicated. When analysing the different aspects mentioned in the previous section, these overlaps have to be understood.

Product, process and assurance case are strongly related. A variation in the product has ripple effects on the process and the assurance case. To exemplify these dependencies at a higher level, Figure 3 is used. This figure illustrates a 3D space populated by families of items. These families represent assurance cases, processes and product-related artefacts.

The Quadrant Z Y of Figure 3 illustrates a product-based fragment of a family of assurance cases related to an automotive family of systems. The presence of the octagon reveals the condition at the variation point, represented by the diamond. Variation points are those places where a choice has to be made. Quadrant X Y illustrates a fragment of the usage context (expressed as a feature diagram) of the family of systems. The usage context also exhibits a variation point (an exclusive alternative between truck or bus). In the context of Figure 3, Quadrant ZY, the choice related to which argument fragment should be selected to continue the argumentation and achieve G3, depends on another choice (represented by the alternative in the feature diagram) in the usage context. Quadrant Z X illustrates a fragment of a family of processes. Even if not explicitly represented, interdependences exist. A choice taken at a variation point within a family of products (e.g., a choice related to the criticality level) may have an impact on the stringency of the process and the argumentation fragment to show process compliance.

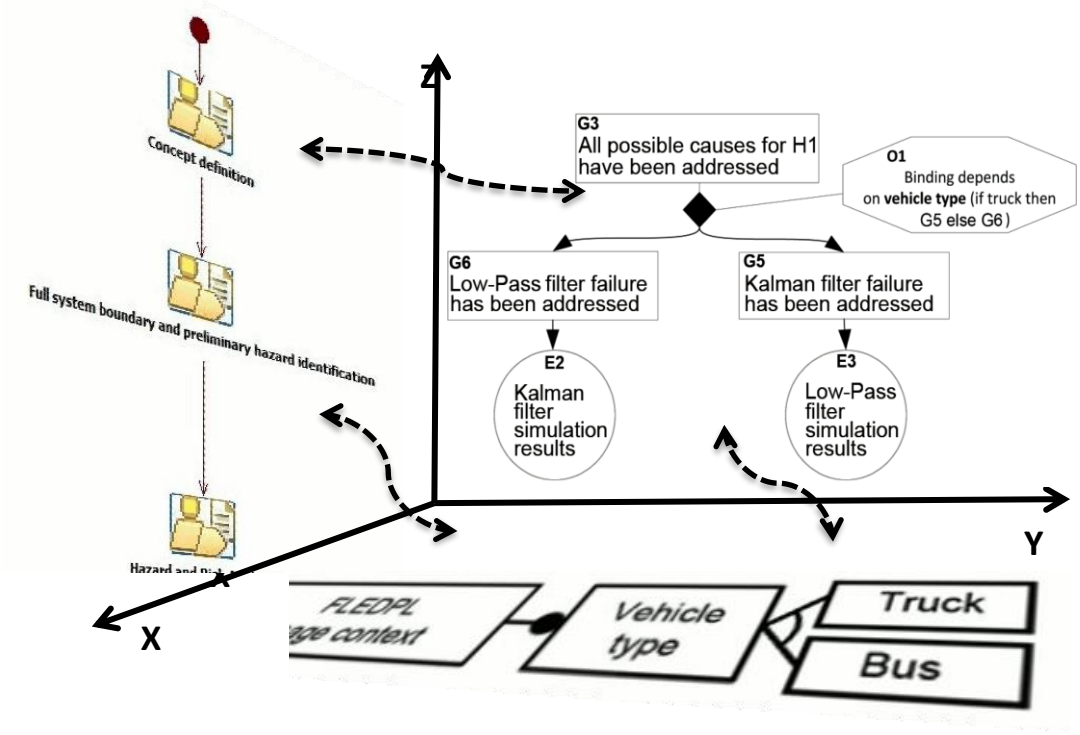

Figure 3: Dependencies between processes, products, and assurance cases, taken from [31]

2.3 Compliance with Standards

As stated in Figure 1, WP6 is particularly concerned with the development of the "Compliance Management" basic building block as part of the AMASS platform. This module is not exclusively concerned with the Reuse topic, but it is a cornerstone aspect for enabling reuse in the safety-critical systems domain.

Rushby [76] states that current certification practice is "standards-based". Standards-based approaches recommend or prescribe a set of objectives, processes and evidential documentation. In AMASS, by compliance we mean the extent to which developers of safety-critical systems have acted in accordance with the "practices" set down in the standards. More narrowly, we can consider this as consistency between the actual development/assurance process and the "normative" models prescribed in the standards themselves.

A principal challenge for AMASS is that currently the existing standards across the avionics, automation, space, railway and automotive domains exhibit wide variation, partial inconsistency, and semantic discrepancy in terms of treatment of common assurance and certification concepts. As a consequence, the provision of a means for the reuse of certification artefacts in the following target circumstances, which form the primary goal of WP6, might be hindered:

- ***Recertification -Reuse of approvals from one application domain to another:*** Reuse of approvals (certifications, assessments, evidence, etc.) in order to reduce time/efforts when we want to certify/assess a product with respect to a domain-specific standard that has been already approved in an application domain with respect to a standard from another domain.
- ***Reuse of approvals within a domain:*** Reuse of approvals (certifications, assessments, evidence, etc.) in order to reduce time/efforts when we want to certify/assess a product with respect to a domain-specific standard that has already been approved in the same application domain with respect to another standard from the same domain, but with a different scope or corresponding to another version of a standard.

- *Note that at a first glance, this last circumstance might be considered remote, contingent. Moreover, since transitional arrangements are usually specified by the new standard/update, this condition might even be considered easier to tackle with respect to the former. Recent experience in the automotive domain shows that transitional arrangements are not always at disposal and thus the introduction of a new standard within the same domain can represent a true challenge.*

The engineer needs to interpret the requirements and objectives of the standards which will apply to the specific situation, and sometimes this is open to interpretations. In order to deal with this problem different guidelines and complements have been published.

Beyond the interpretation of standards, when trying to analyse the different standards, it is beneficial to create a common framework so that the peculiarities and commonalities of the standards can be highlighted. Among other approaches, the OPENCOSS and SafeCer projects focused on creating such a common framework in order to compare standards from various industrial domains (e.g., avionics, automotive, railway and medical devices). These approaches facilitate the reuse of process-related assets. In Section 3.1, we investigate them, their limitations and their usage for reuse purposes.

3. State of the art on cross-and intra-domain reuse

Cross- and intra-domain reuse embraces immediate, direct and indirect evidence. Thus, it embraces architectural artefacts and V&V artefacts (both related to the expected work on architecture-driven assurance), process-related artefacts, as well as multi-concern artefacts in terms of cross-concern reuse (related to the expected work on multi-concern assurance). Thus, reuse is a transversal objective. This chapter provides an overview concerning the state of the art on cross and intra domain reuse. The overview is taking into consideration different foci. The main foci are: process-based reuse, product-based reuse, and cross-concern-based reuse.

For each of these foci, different methodological solutions are discussed. These solutions embrace: pattern-based, family-oriented, component-based, and model-based solutions.

3.1 Process-based reuse

In this section, we discuss process-based reuse. More precisely, reuse of safety/security plans; reuse of process tasks/steps; and reuse of organizational structures. More in general, we overview the reuse of process-oriented evidence (which is typically classified as indirect in the scientific literature, as mentioned in Subsection 2.1.3); reuse of process-oriented argumentation artefacts (e.g. safety audits); and systematic/non-systematic approaches.

3.1.1 Compliance with processes prescribed by standards

As defined in [78], "*standards are documented agreements containing technical specification or other precise criteria to be used consistently as rules, guidelines, or definitions of characteristics, to ensure that materials, products, processes and services are fit for their purpose*".

Dodd and Habli [77] distinguish two types of certification standards: prescriptive standards and goal-based standards. In the first one, applicants show that the system is acceptably safe, based on the achievement of the process objectives prescribed by the standards. Goal-based standards, on the contrary, request the existence of a clear argumentation relating how evidences generated as an output from testing, analysis and review, support claims concerning the safety of the function. For instance, IEC 61508-3 follows the prescriptive certification approach while ISO 26262 exhibits a hybrid approach: on one hand it requires the creation of a safety case, but on the other hand the safety case itself is defined as compilation of work products generated during the development process.

The standard-driven way of ensuring quality is pursued by imposing a (criticality-dependent) level of rigor on the processes and workflows used to build the final system and by specifying the intermediate artefacts to be produced.

To make compliance tasks more cost-effective and more reusable, there have been some initiatives working in defining ways to: (a) specify standards and related information in a structured way, (b) store the standards in a form that can be retrieved, categorized, associated with other standards, searched and browsed, (c) manage the demonstration of compliance with standards by providing means to categorize evidence information.

Emmerich et al [79] designed a model for managing standards requirements for compliance management. This model has two parts:

(1) standard specification and its use,

(2) activities performed during the project execution to evaluate compliance level.

The “document” becomes the core artefact to be checked. Any document can be in one of these states: compliant, noncompliant, unsafe, check not required (at a given time) or undefined. Based on these states, the compliance level can be easily obtained during development time.

The EVOLVE (Evolutionary Validation, Verification and Certification) project [80] created a compliance framework based on a structured semi-formal meta-model. This meta-model helped to build domain-specific libraries of standards models (IEC 61508, ISO 26262, DO-178C). The meta-model allows users to set guidelines, similar to “spell-checking”, in which a number of compliance checks are performed to assess the degree of compliance of embedded system products against industry standards.

The ModelME! (Model-Driven Software Engineering for the Maritime and Energy Sectors) project [81] defined a meta-model to ensure compliance with IEC 61508. The conceptual model defined in the ModelME! project provides an overall view of the safety evidence pieces and the interconnections that need to be established between these pieces during the lifecycle of a safety-critical system. This fact relates the ModelME! model to a prescriptive type of argument focused on standard requirements (detailed in Requirements Concepts package in the model itself).

Chung et all [82] defined a meta-model of standards to support IEC 61508 processes, requirements and lifecycle. A User-defined Process (UDP) can be also modelled using the meta-model. The Compliance check is made by an inspector that verifies the accomplishments of IEC61508 requirements against the UDP. Compliance management in this approach includes: (1) identifying the corresponding specification and tasks against the standard; (2) correctness check to verify that the placement of task specifications comply with the standard requirements; (3) capability check to ensure that developers comply with the required capability; (4) recommendation check to ensure that techniques, measures, tools and methods prescribed by the standard are considered in the UDP; (5) planning assistance, for providing a guideline for defining UDP according to IEC61508 defined requirements, tasks, capabilities, techniques, and (6) cross-referencing, for managing compliance from standard to UDP and vice versa.

The OPENCOSS project [83] defined the CCL (Common Certification Language) meta-model as a basis to specify functional safety standards, assurance concepts, and projects, in a well-structured, well-ordered and, hence, comparable way. The meta-model is holistic and generic, and it abstracts common concepts for demonstrating safety compliance from different standards and application domains. Its application results in the specification of “reference assurance frameworks” for safety-critical systems. Reference assurance frameworks correspond to a model of the safety criteria of a given standard.

There is definitively quite significant work in specifying: standards, interpretations, and means of compliance, in a structured way. There are some limitations on these approaches, which are part of the AMASS challenges:

- Practitioners need to determine the assurance objectives to be reached and the process to be executed, based on the characteristics of a particular system. As the text of the standards can sometimes be ambiguous, inconsistent, and hard to understand, this can become an arduous task.
- We must support compliance demonstration to diverse regulatory standards while addressing heterogeneous processes defined for diverse domains, such as automotive, automation, space, railway or avionics. Other domain such as medical and nuclear, even if not directly targeted by AMASS, are also expected to benefit from AMASS-expected results concerning compliance management. These mandated processes are not comparable. They must be defined in different ways, at different granularity levels, and even for different targets/goals/outputs. For a given development in a certain domain, the applicability for conformance in a different domain could not be straightforward. This presents particular challenges to the reuse of compliance assets across various standards and domains.
- Demonstration of compliance with standards becomes even more difficult when a system changes. For example, evidence evolves when a system aims at being certified against different standards or

reused in another application domain. Although the correspondence between standards was studied, it is a complex task. No perfect match usually exists between the compliance needs of different standards, and system suppliers usually have their own interpretations and thus usage of each standard. As a result, compliance with a new standard is never straightforward.

3.1.2 Family-oriented solutions: Process-line engineering

To enable reuse within a community of practices, a process line-oriented approach might be beneficial. A process line is defined as a set of similar processes within a particular domain or for a particular purpose, having common characteristics and based upon common and reusable process assets [35]. Via a process line-oriented approach, it becomes possible to systematize the commonalities and variabilities among practices. In recent years, different mechanisms have been proposed to manage the variability of process lines and to generate processes that satisfy requirements of specific projects. We can find two types of approaches, constructive approaches and evolutionary approaches [36]. While constructive approaches focus on defining process variants by means of models that contain all the variability of the process. Evolutionary approaches focus on the evolution of reference processes using different operators. In this section, we take into account the most remarkable works in these two lines of research.

The well-known OMG standard Software & Systems Process Engineering Meta-model (SPEM) 2.0 [37] for process modelling is an example of a constructive approach for process lines. It includes variability mechanisms (see Figure 4) that can support process-line engineering approaches. Specifically, SPEM 2.0 defines the variability between two process elements (i.e. work products, roles) of the same type using four types of relationships that define the way in which one element (the variant) modifies the other (the base). The possible relationships considered are *na (meaning not assigned)*, *contributes*, *replaces*, *extends* and *extends-replaces*.

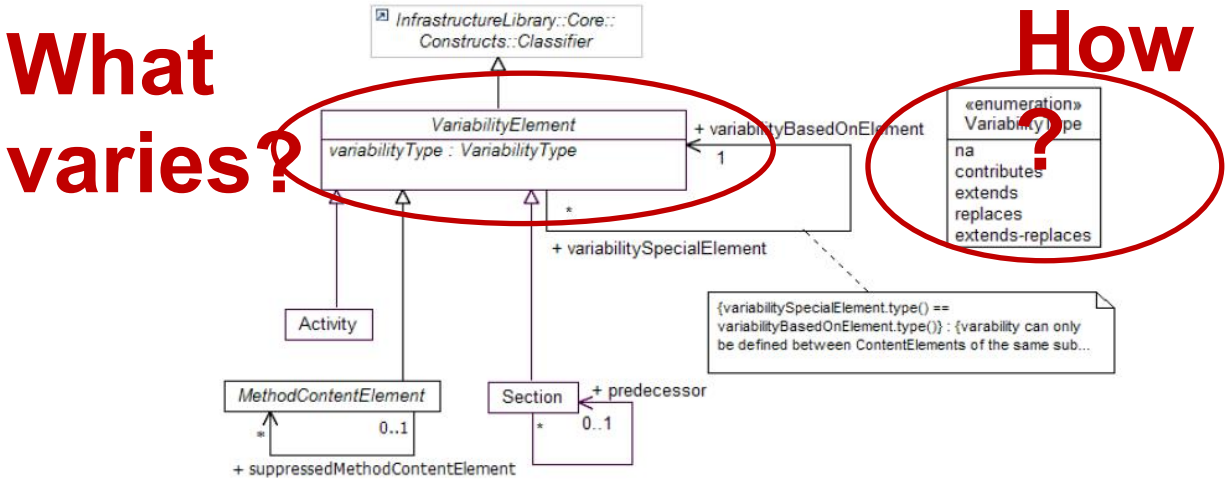

Figure 4: Variability classes in SPEM 2.0 [37]

SPEM 2.0 presents two main limitations to model the process variability [38][39][40]. Firstly, the variability modelling is restricted to elements of the Method Content package, so the variability is only considered in modelling of the process and cannot be exploited during its execution. Secondly, the variability is expressed at the level of the element of a process, so for complex processes, in which many process elements can take part, it is hard to predict how variability relations interact with each other. Several works have been proposed in order to overcome these limitations by proposing extensions of the SPEM 2.0 meta- model or providing tools that reduce these limitations.

One of the most remarkable extensions of SPEM 2.0 to support process line engineering is vSPEM [39]. This proposal extends the SPEM 2.0 meta-model (see Figure 5) with concepts of the Software Product Line

domain [40]. Specifically, it includes the concept of *variation point*, i.e. a process element that can vary, and *variant*, i.e. alternatives provided to a variation point. The *occupation relation* (depicted in the bottom of Figure 5) models the relationship between variation points and variants. One variation point can be related to one or more variants by means of an occupation relation. The vSPEM extension has been validated with real users [41] demonstrating that its variability mechanisms exhibit a greater understandability than the original SPEM 2.0 mechanisms. However, the vSPEM process diagram understandability is lower than that of SPEM 2.0. One of the main limitations of vSPEM is, that there are no tools that support its notation and mechanisms to model and resolve variability.

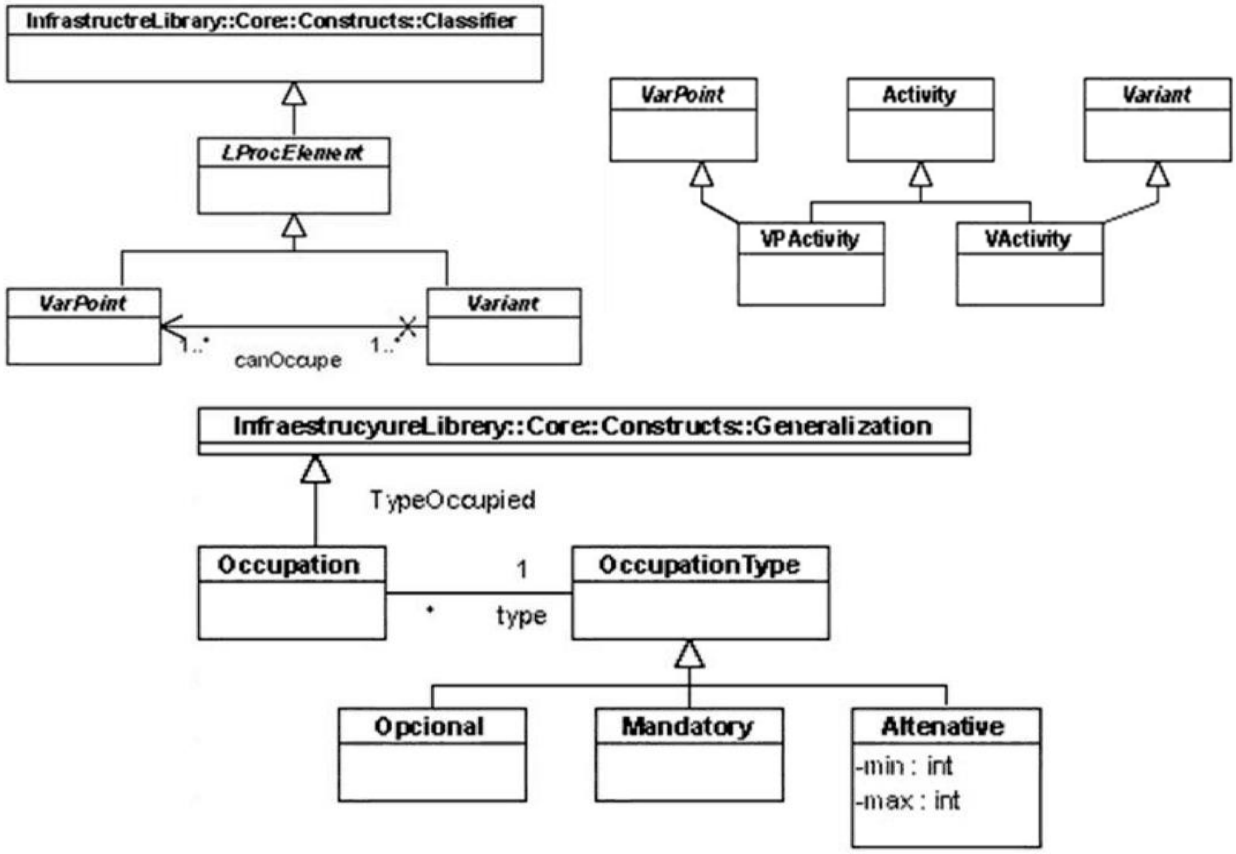

Figure 5: Variability extensions of vSPEM with respect to SPEM 2.0 [39]

The work presented in [40] studies the integration of vSPEM with well-known product line notations like feature models [41] and their tools. The conclusion of the study is that the presented tools can be used to analyze the process line and check properties like the correctness and the number of processes that can be generated from the process line.

SMartySPEM [42] is another extension of SPEM 2.0. SMartySPEM offers a set of guidelines for the identification and management of variability in Software Process Lines. SmartySPEM is implemented as an extension of the UML profile of SPEM 2.0, so current implementations of the profile can be re-used to model the variability according to SMartySPEM. This profile extension adds new structures for modelling the variability of the process line that are not in vSPEM, like the concept of inclusive or exclusive variants and constraints between variants of the system. However, it does not support the automatic derivation of specific processes.

Some process lines proposals use already available tools of the Software Product Line community to manage the variability of the process line and to generate tailored processes. In this line, F. A. Aleixo et al. [43] presents a process for the variability management, customization and execution of software processes based on pre-existing tools (see Figure 6). In this proposal, the process line is modelled using the Eclipse

Process Framework (EPF) composer [44] and the management of the variability and the customization of the process are made in a modified version of the GenArch tool [45]. Finally, a set of ATL transformation rules are used to transform process models provided by GenArch in a process that can be executed in the JBoss BPM workflow engine [46]. Rouillé et al. [47] present a work that also relies on Software Product Line tools, but using the proposed OMG standard for variability, called CVL (Common Variability Language)[2], and its associated tool [66]. The main advantage of these proposals is that they rely on standard meta-models for the specification of processes. They augment the capabilities of SPEM to manage variability without modifying its meta-model. However, the problem of the work presented by F. A. Aleixo et al. is that it uses tools that are not available or unsupported, and this makes its application to practical cases very difficult. The work presented by Rouillé et al. can be applied instead, being supported by a new version of the CVL tool, known as the BVR tool [67], an output of the European project VARIES [68].

Figure 6: F. A. Aleixo et al. approach

CASPER [48][49] is an example of an evolutionary approach for process lines. This proposal uses Model Driven Engineering (MDE) and process lines for process tailoring in the context of small and medium enterprises. Authors propose a MDE-based approach for software process line analysis and design, focusing on process tailoring with respect to a context. This proposal uses two models as input, a software process defined in a simplified variant of SPEM 2.0 (eSPEM) and a model of the organizational context of the enterprise for a specific project (see Figure 7). Then, using a set of ATL[3] transformation rules [50], a new adapted process model is generated. The key idea of the authors is to codify the knowledge of the process engineer to tailor a process by using ATL transformation rules according to the context. This approach has been validated in the context of two small enterprises. The main advantage of this proposal is that it relies on ready-to-use technologies, so it is relatively easy to re-use results provided by this work. On the other hand, the definition of the transformation rules could be very complex even for simple processes and context models. Due to this, authors target this proposal to small companies only, linked to a particular niche having a formalized software process in place.

[2] CVL is a Domain Specific Language (DSL) for modelling variability in any model of any DSL based on Meta-Object Facility (MOF).

[3] ATL stands for ATLAS Transformation Language, a hybrid model transformation language that allows both declarative and imperative constructs to be used in transformation definitions.

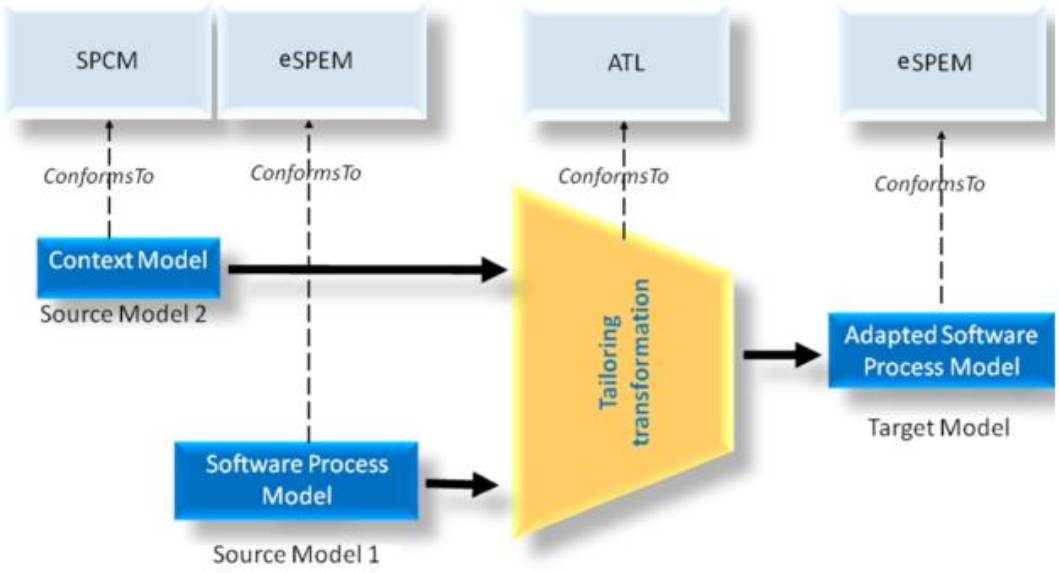

Figure 7: The CASPER approach [43]

The V-Modell XT model [51] is the standard software process for IT development projects in Germany's public sector. It supports software process variability using an evolutionary approach. This model supports a great variety of variability operation types, which comes from the renaming of a specific role to removing a task (see the complete list of variability operation types in [52]). J. Schramm et al. [53] explore the feasibility of the variability operations supported by this process model to support the development of flexible software process lines. With this goal, they study the evolution of the five variants of the V-Modell XT process during 2 years. Their conclusion is that variability modelling is a useful instrument to implement variability in real-life software process lines.

In the context of the SafeCer project, Gallina et al. have extended the notion of process line to reason about families of safety processes [28][29]. The extension is called "Safety-oriented Process Line (SoPL)" and the engineering of SoPLs is called SoPLE. Specifically, SoPLE targets safety standards. As Figure 8 shows, during the domain engineering phase, commonalities and variabilities are analysed and systematized, then during the process engineering phase, commonalities and variabilities are reused in order to compose single processes. SoPLE has the potential to be effective to reduce certification costs for enterprises that produce slightly different safety critical systems, or systems that must be compliant to multiple safety standards. The proposal is implemented using the SPEM 2.0 meta-model and the EPF-Composer tool.

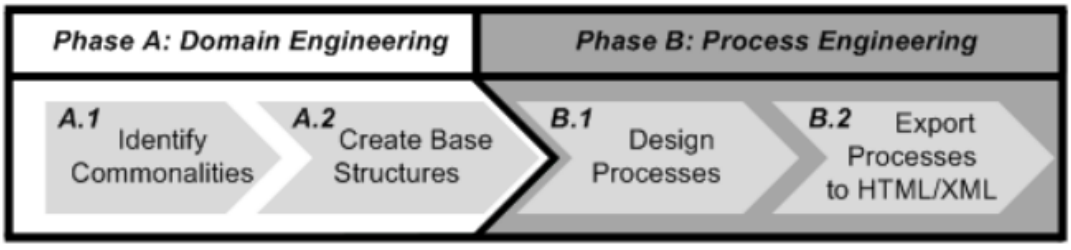

Figure 8: Domain and Application Engineering of the Gallina et al. approach [30]

In the context of SafeCer, metrics to calculate the Reuse Rate for SoPL Process Elements were proposed. Gallina et al. [32] had also pioneered a method called OPER to enable an ontology-based process-elements reuse. Ontologies, representing process-models, could be compared and contrasted in order to merge them within a unique model representing a family of processes, i.e. a process line.

In the context of business process models, process lines and mechanisms to manage their variability have been proposed [54][55]. Here we review the three most remarkable approaches in this domain, C-EPC [56], Provop [57] and PESOA [58]. These works have in common that are well stablished and highly cited work that have a mature tool support.

C. Ayora et al. present an interesting application of C-EPC and the VIVACE framework [56]. The work presented in [54] facilitates the management of the variability presented in process families, while ensuring family correctness and reducing the effort needed for such purposes. The authors identified 10 change patterns derived from a literature review on variability-specific language constructs, which includes the works of Provop and PESOA. Authors explore the feasibility of the proposal modelling the standard IEC 61508. Figure 9 is taken from their exploration-work. Additionally, they compare the effort to model and to evolve IEC 61508 using this proposal with the effort to perform the same tasks using SafetyMet [59], a generic meta-model for representing safety standards. The results of this comparison show that the application of the patterns can reduce the number of operations when modelling a process family by 34% and when evolving it by 40%.

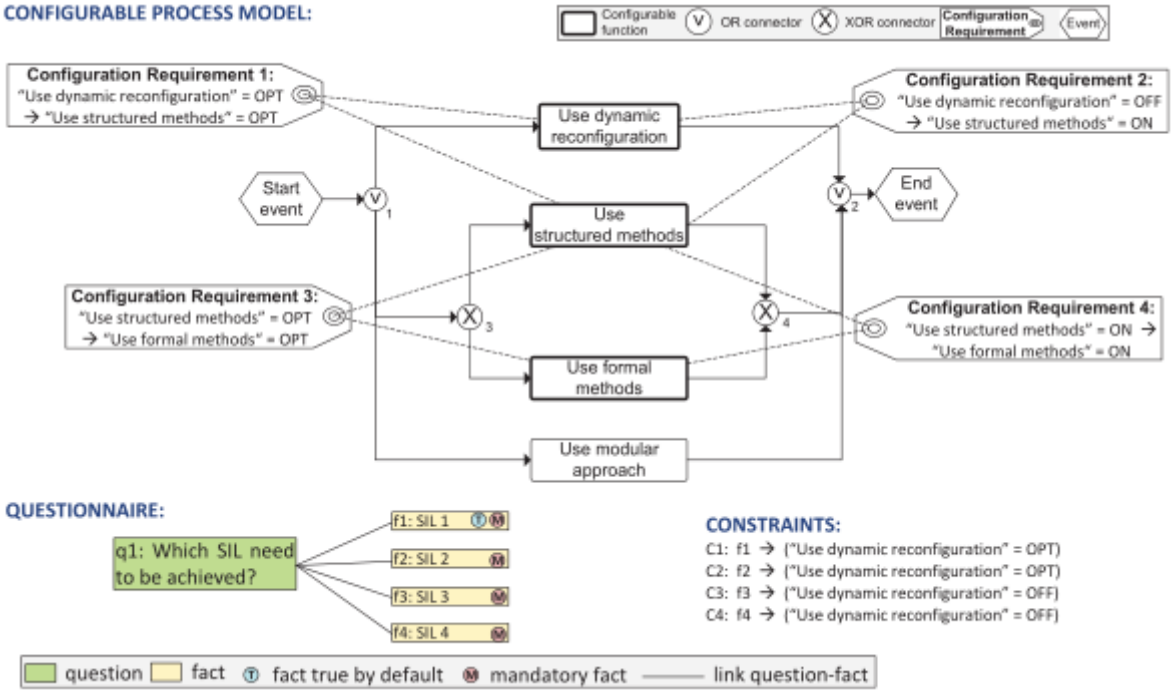

Figure 9: Modelling of part of Standard IEC 61508 in C-EPC [51]

The goal of the Provop framework is to model and manage large collections of business process variants. The approach of Provop is similar to the work presented in [56], so it defines a reference process model and the adjustments to configure this process to different process variants in an effective and manageable way. In the work [60], A. Hallerbach et al. provide a proof-of-concept of Provop using the ARIS Architect tool [61] with the BPMN notation. A specifically developed ARIS script generates the variant of the base process.

The main goal of the PESOA project [62] is the implementation of a process family engineering platform. Their proposal is based on the use of Domain Specific Languages, Feature Models and Generative Techniques. The Hyper-Sense tool [63] fully supports this proposal in an integrated environment with graphical editors. The use of feature models to deal with the variability of Business processes is also approached by Czarnecki et al. in [64], using super-imposed models and UML activity diagrams. In this

approach, control flows of UML activity diagrams are annotated with presence-conditions and meta-expressions linked to feature models for modelling if an element is going to appear in a variation of the activity or not. An Eclipse plugin [65] implements this approach, however, it has not been validated in real scenarios.

Zeller et al. [86] propose a cross-domain assurance process in conjunction with a development methodology for safety-relevant software. The objective was to reduce the effort required to perform a safety assessment by reusing safety analysis techniques and tools as well as artefacts produced during the safety assurance process. The process consists of generic and domain-specific steps that must be executed in each of the considered domains as well as steps that are only necessary in specific domains. The authors were able to reuse techniques and tools for safety analysis on different domains. However, not all of the phases of their proposed process were domain-independent, and safety-certification artefact reuse was not considered in their research. Papadopoulos and McDermid developed a similar approach [87].

3.1.3 Process-based patterns of reusable reasoning

MDSafeCer [26] is a remarkable proposal of process-based patterns of reusable reasoning. As shown in Figure 10, the process-based pattern, given in GSN [118], in MDSafeCer is constituted of a top-level claim stating that "the adopted process is in compliance with the required standard and integrity level". This claim is expected to be decomposed by showing that all the activities have been executed and that in turn for each activity all the tasks have been executed until an atomic process-related work-definition unit is reached. A more detailed version of this pattern is given in [26]. Figure 10 reproduces a portion of such a pattern.

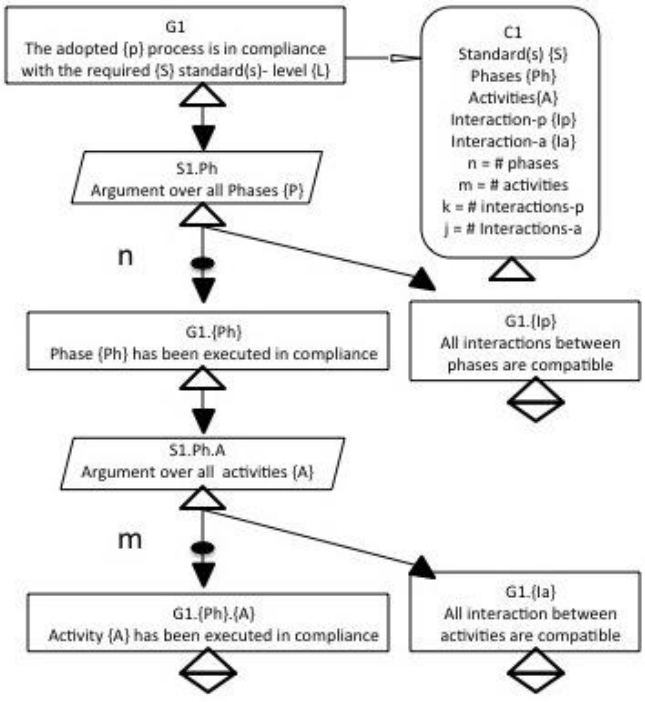

Figure 10: Goal structure representing Process compliance argumentation pattern, taken from [26]

Nair et al. [69] recognize the relevance of process-based argumentation and similarly to what is proposed by Gallina [25], they argue about the core process elements. Nair et al. call the process-based argument as secondary confidence argument.

A pattern to argue about an entire family of processes was developed in SafeCer and presented at DEVVARTS [27]. Figure 11 reproduces its main reusable reasoning, given in GSN [118].

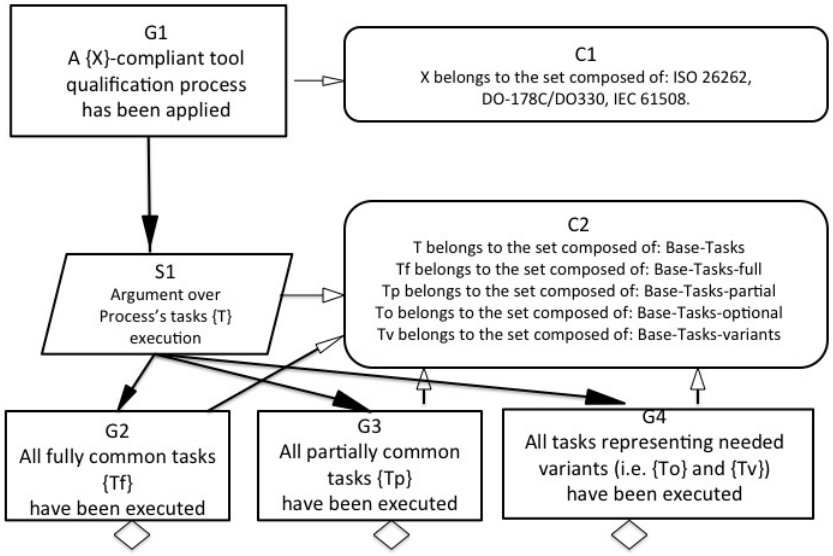

Figure 11: Goal structure fragment representing a process line-based argumentation pattern

3.1.4 Process-related Reuse based on Cross-Mapping of Standards

This topic particularly relates to cross-standard reuse (standards likely from different domains), in which a previously certified product typically needs to be re-assessed against different standards. Unlike the process-line engineering approach, the cross-mapping approach does not involve the creation of common process models to be instantiated by resolving the variability points. Instead, this approach defines the equivalence mappings between different models of standards/processes in order to identify the reuse opportunities.

The OPENCOSS project has followed this approach. As described in Section 3.1.1, the reuse across standards and domains builds upon the CCL meta-model. Figure 12 sketches the CCL meta-model packages.

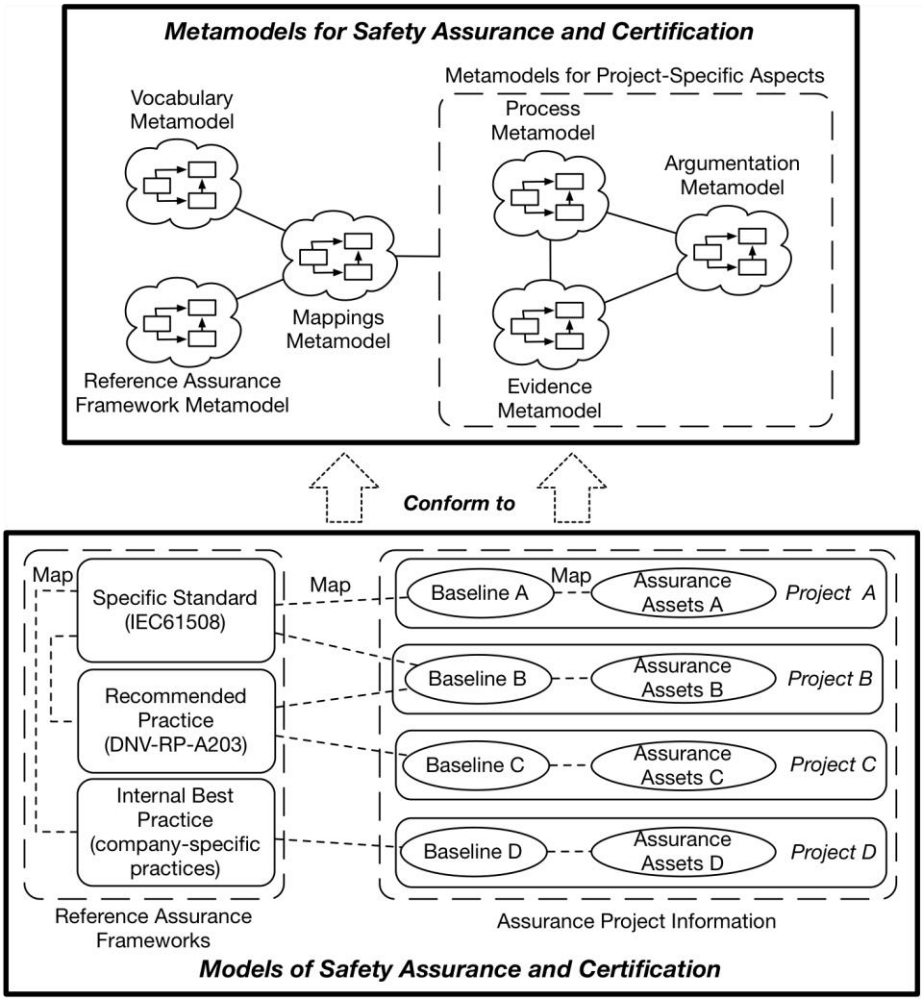

Figure 12: Overview of the OPENCOSS CCL meta-models for assurance and certification

- The Reference Assurance Framework Meta-model supports the specification of the compliance requirements that have or might have to be considered in an assurance project. Compliance requirements can be from specific standards, recommended practices, or company-specific practices, and typically have to be tailored to project-specific characteristics. The latter is done by means of baselines, which correspond to the specific safety criteria of a standard with which a given assurance project has to show compliance. A baseline is usually represented by a subset of all the safety criteria required in a standard, and it varies according to projects. For example, the safety criteria will vary if a system is developed using model-based techniques.

- Another source of information for compliance is the data about the product for which compliance is sought. Therefore, the meta-models also include the concepts and relationships necessary for modelling and managing project- and product-specific information. This information is referred to as assurance assets and needs to be recorded regardless of which standard is being followed. OPENCOSS has defined meta-models for modelling the assurance assets of a project in the form of:
  - The process executed to create a product (Process Meta-Model);
  - The evidence for compliance (Evidence Meta-Model), and;
  - The arguments that will be used to justify key safety-related decisions and compliance (Argumentation Meta-Model).

  The evidence, in the form of artefacts, can be both input and output of the activities of the process and can support the arguments. The arguments can refer to process aspects.

- The Vocabulary Meta-Model is a means to define and record the terms and concepts used to characterize reusable assurance assets such as evidence, argumentation, and process data. The terms of the vocabulary can thus be used when naming or describing the assurance assets, as well as elements of a reference framework. Terms and concepts of the vocabulary can be specified from the text of a standard or be specific to some company, product, or application.
- By using the Mappings Meta-Model, maps can be created to specify the degree of equivalence between vocabulary terms (e.g. from different domains), between the assurance information gathered during a project (e.g., artefacts) and its baseline for indicating compliance, and between standards (i.e. reference assurance frameworks) for indicating how the standards relate. The latter is a key to reuse safety certification artefacts across different standards and domains. In general, the mappings allow engineers and managers to make informed decisions about the appropriateness and implications of re-using assurance information across projects, standards, and domains.

3.1.4.1 Principles for reuse of artefacts across standards and domains

According to the OPENCOSS analysis, there are four main principles that should be taken into consideration when reusing artefacts across standards:

1) the intent of a safety certification artefact must be taken into account when aiming at its reuse,
2) maps must be established between the source standard (*reuse from*) and the target one (*reuse to*),
3) project compliance must be determined (by means of maps), and
4) needs and gaps resulting from safety certification artefact reuse must be determined. These four principles are orderly described in the following.

1) Certification artefact intent

Certification artefact reuse is not a challenge per se. In theory, any artefact is reusable. The main reuse need stems from the fact that each standard has its own requirements to fulfil, and such requirements can vary among standards. For example, DO-178C lists objectives (requirements) for the different software development processes, and some of its objectives are not fully addressed in EN 50128. When reusing a certification artefact produced in compliance with a source standard, it must be determined which requirements of the target standard are fulfilled. Certification artefact reuse can be regarded as the process targeted at determining which requirements of a given (target) standard are fulfilled when compliance with another (source) standard has been achieved.

The above need can only be met if the standard's requirements, whose fulfilment a certification artefact contributes to, are recorded. These requirements correspond to the artefact intent: which properties are assured in the artefact, and thus why the artefact is necessary. For example, the DO-178C Software Requirements Data must include the performance criteria, timing requirements and constraints, and memory size constraints so that the artefact fulfils its intent (i.e., to show that such characteristics have been considered and specified).

The certification artefact intent is also based on the activities that use or produce the artefact. In general, and considering that activities use and produce several certification artefacts, the overall aim of an activity corresponds to a higher-level intent than the individual intent of the output artefacts of the activity. The achievement of this higher-level intent is also enabled by the individual intent of the input artefacts of the activity. For example, EN 50128 Integration Process (activity) aims at demonstrating that software and hardware interact correctly to perform their intended functions. To this end, the activity uses the Software Integration Test Specification and the Software/Hardware Integration Test Specification as input, and produces the Software Integration Test Report and the Software/Hardware Integration Test Report as output.

2) Equivalence mapping between standards

In addition to recording the intent of the certification artefacts, it is also necessary to determine the equivalence between standards for certification artefact reuse. This can be done by means of maps that indicate the extent to which the requirements of the standards (e.g., requirement concerning the provision of a certain artefact) correspond (e.g., between EN 50128 Software Requirements Specification and DO-178C Software Requirements Data). Based on these mappings, the similarity and differences of the standards can be assessed, and thus how compliant a certification artefact is with respect to a given (target) standard according to its compliance with another (source) standard.

Three general types of maps can exist between the elements of two standards:

- ***Full map***: the elements of the two standards are identical; the characteristics of the element in its original context (its form, required content, preconditions, objectives, post-conditions on use...) fully satisfy the requirements of the context in which it is to be reused.
- ***Partial map***: the elements are similar, but they are not identical; depending on the context and the objectives, the differences between them might be significant; in this case, a clear record of the similarities and differences is required.
- ***No map***: there is insufficient similarity between the elements to enable us to assert a mapping; in this case, it may be important to record the differences, and the reasons why the mapping makes no sense, in order to spawn further gap analysis and prevent inadvertent reuse.

Full maps are usually rare in the assurance domain and the majority of maps are partial.

Three elements play a role in equivalence mapping: artefacts, activities, and requirements. Pragmatically, any of these referenced assurable elements can be reused. The acceptability of the reuse needs to be argued in terms of the overall assurance objectives indicated by a standard: i.e. what needs to be demonstrated for assurance and compliance in the target context.

Equivalence maps are also necessary between the baseline of an assurance project and the reference framework (or frameworks) of the standard according to which a system has to be assured. Some differences might exist as result of e.g. having to tailor how to follow a standard according to the specific characteristics of a system.

3) Compliance mapping

Another necessary type of map for cross-standard and cross-domain reuse is compliance map. These mappings specify how the information of an assurance project (i.e., its body of assurance assets) complies with its baseline. As for equivalence maps, compliance maps can be full, partial, or no map. By mapping an artefact to a reference artefact selected for a baseline, the intent of the artefact has to be indicated.

The compliance maps of the source assurance project will typically be full and 1:1. Its baseline will correspond to a template which is used in the project. For example, a baseline from a reference framework for DO-178C will have Software Requirements Data as an artefact to provide, and an assurance project can have a single artefact that maps to Software Requirements Data. Nonetheless, an assurance project can also manage, structure, or group its artefacts in a different way as a standard indicates, but still being compliant. For example, an assurance project could have more than one artefact for its Software Requirements Data, such as high-level requirements specification and low-level requirements specification. Each of these artefacts would partially map to Software Requirements Data.

Compliance maps for the target assurance project can be derived from the compliance maps of the source project. In this case, the likelihood of derived full maps is low because of the differences that usually exist between standards. The assurance information of the source project fulfils the requirements of its baseline and this baseline has been generated from a specific reference framework (both belonging to the source

domain for the cross-domain reuse). The target assurance project will have a different baseline and reference framework, thus different requirements to fulfil. Nonetheless, some reference requirements can be similar or equal. For example, an artefact that complies with EN 50128 Software Requirements Specification will partially map to the DO-178C Software Requirements Data.

4) Needs and gaps from certification artefact reuse

Finally, reuse of safety certification artefacts across standards and across domains requires that compliance needs and gaps resulting from the reuse are determined. This is a consequence of the fact that the maps between standards are most often partial. Compliance with a standard will allow a system to comply with another to some extent, but fulfilment of further reference requirements for the target project may be necessary. This will usually involve the execution of some additional activity and possibly the creation of some additional artefacts.

For example, reusing EN 50128 Software Requirements Specification in an avionics assurance project will lead to the partial fulfilment of the reference requirements for DO-178C Software Requirements Data. The corresponding compliance gaps must be filled for the achievement of full compliance with DO-178C.

3.1.4.2 Process for reuse of assurance assets across standards and domains

The process consists of seven main activities and it is based on the conceptual framework shown in Figure 13. This framework corresponds to excerpts of all the meta-models created in OPENCOSS [83]. Although presented as a sequence, some activities of the process can be executed in parallel, iteratively, or in a different order. For example, the baseline of an assurance can be modified while collecting new assurance assets because of some new request by a safety assessor.

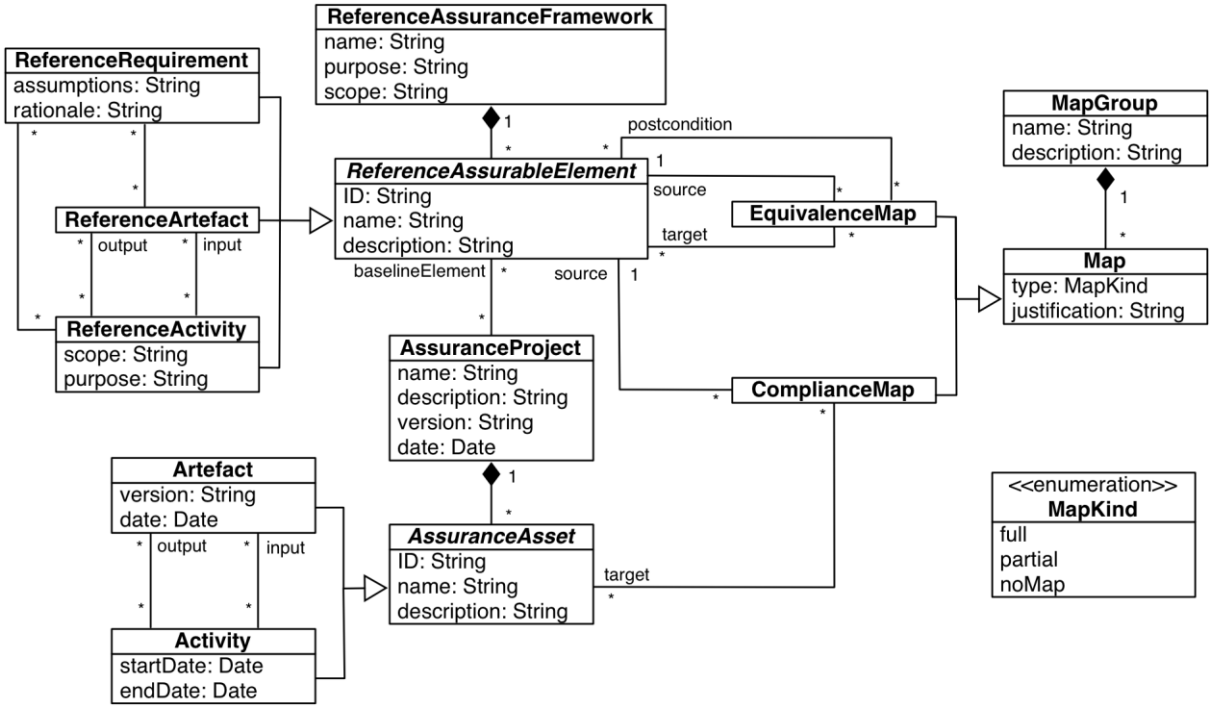

Figure 13: OPENCOSS reuse conceptual framework

The seven main activities are described below.

1) Create reference assurance frameworks

The first activity to be executed to apply the reuse approach is to specify the compliance needs of the standards to comply with the source and the target domains. A reference assurance framework must be

created for each standard, and the frameworks will be later used for specifying baselines and equivalence maps.

The reference frameworks will contain reference requirements to fulfil, reference activities to execute, and reference artefacts to manage. Reference artefacts can be linked to reference requirements and to reference activities (input/output), which enables the specification of reference artefact intents. In EN 50128, Software Design Specification is a reference artefact, Component Design is a reference activity, and "test cases and their results shall be recorded, preferably in machine readable form for subsequent analysis" (clause 7.6.4.5.a) is a reference requirement for the Software Integration Test Report. Reference artefacts, activities, and requirements can be decomposed into others.

A reference framework can further contain information about the reference roles that might be involved in a safety-critical system's lifecycle (e.g., designer), about the reference techniques that might be used to execute reference activities and create reference artefacts (e.g., formal methods), and the applicability of the above elements (e.g., a given reference technique can be recommended for a given SIL in EN 50128). A reference artefact can also have reference artefact attributes (e.g., test outcome; passed or failed) and be linked to other reference artefacts by means of reference artefact relationships (e.g., Design Description satisfies Software Requirements Data).

2) Specify equivalence maps between the reference assurance frameworks

Once reference assurance frameworks are created, equivalence maps can then be specified between their reference assurable elements in order to determine how similar the frameworks are. Each equivalence map will have a source and a target reference assurable element, and can have a textual justification, and post-conditions. The post-conditions correspond to additional reference assurance elements that must be taken into account when an artefact is reused.

When creating equivalence maps for safety certification artefact reuse (i.e., between reference artefacts), and as explained above, it is essential to analyse the intent of the reference artefact to decide upon the extent to which two reference artefacts are similar. Arguing about the (relative) equivalence between reference artefacts requires discussion of the similarity of the associated reference requirements and reference activities.

The equivalence mapping process might not result in 1:1 maps between e.g. single reference artefacts. A set of source reference artefacts might map as a whole to a single target reference artefact. It can also be necessary to map different types of reference assurance elements. For example, there may be cases where no mapping can be asserted between a reference requirement of the target reference assurance framework and any of the reference requirements of the source, but that the former reference requirement could be mapped to some reference artefact in the source. For the two cases described in this paragraph, the specification of a map justification is especially important to document why the corresponding maps have been specified.

Another aspect to take into account is map consistency. For example, if the reference requirements of a reference artefact only partially map to the reference requirements of another reference artefact, from a different reference assurance framework, then the map between the reference artefacts must be "partial". As an example, EN 50128 Software Requirements Specification partially maps to DO-178 Software Requirements Data because their reference requirements are different, thus what can be assured with them.

3) Determine the baseline for the source assurance project

The specific compliance needs of the source assurance project must be specified. To this end, the applicable elements of a reference assurance framework must be selected. It is also possible to refine the

elements to project-specific needs or to specify information that it is not in the reference assurance framework. For example, DO-178C does not explicitly define roles for assurance projects, but such roles might have to be declared for a specific project. When refining a reference assurance framework into a baseline, the suitability of the changes introduced for the corresponding assurance project, in relation to how the baseline corresponds to the reference assurance framework, might needed to be justified. This is achieved by specifying equivalence maps between the baseline and the reference assurance framework.

4) Collect the assurance assets of the source assurance project

The information of the assurance assets of the source assurance project must be collected in order to demonstrate how the system safety has been assured. For reuse purposes, the main assurance assets to collect are the artefacts created during the system lifecycle. Other assurance assets include the information about the activities executed, the participants in the assurance project, the techniques used, and the argumentation claims.

5) Specify compliance maps between the assurance assets of the source assurance project and its baseline

Compliance maps must be specified for the source assurance project in order to indicate how the project meets its compliance needs. The assurance assets of the project are mapped (full or partial map, or no map) to its baseline, and a justification for the map might be specified.

6) Determine the baseline for the target assurance project

This activity is the same as (3) but for the target assurance project instead of the source one.

7) Reuse assurance assets from the source assurance project to the target one

Finally, assurance assets from the source assurance project are reused in the target one, for the baseline specified in activity (6). The reuse can result in additional compliance needs for the target assurance project according to the post-conditions of the equivalence maps. These needs will be included in the baseline of the target project.

Reuse of assurance assets results in the generation of compliance maps for the target assurance project. This is based on the chain of maps consisting of the compliance maps for the source assurance project and of the equivalence map between reference assurance frameworks, including possible maps between a baseline and a parent reference assurance framework. If there is some partial map in the chain, then the resulting compliance map for the target assurance project will also be partial. It might also be necessary to address some post-conditions, and it is also essential to analyse the map justifications in order to determine what compliance needs must still be addressed in the target assurance project. When all the maps are full, instead, then the derived compliance map is supposed to be full too. If there is any ‘no map’, then the reuse is not advisable. This means in practical terms that no reuse is possible, unless at the expense of significant reverse engineering activities, almost as it were a new project.

For assurance asset reuse between projects, mapping the reference requirements can suffice because the reuse consequences can be determined from the correspondence between reference requirements. These consequences can correspond to the need of assuring further reference requirements in the target assurance project.

3.1.5 Model-based process compliance

Meta-modelling in the context of assurance cases is a rather recent research topic, and thus the development of model-driven methods has been delayed due to the absence of standardized, stable and fully formalized meta-models. The development in this direction, as carried out within SafeCer, is recalled in what follows.

MDSafeCer (Model-driven Safety Certification) [25] is a method that enables the semi-automatic generation of composable argument-fragments within safety cases aimed at arguing about process compliance with respect to safety standards. Via MDSafeCer, as shown in Figure 14, process models, compliant with a process modelling language meta-model (e.g., SPEM 2.0), are transformed into argumentation models compliant with the Structured Assurance Case Meta-Model (SACM) [120] and presented via, for instance, GSN-goal structures [118].

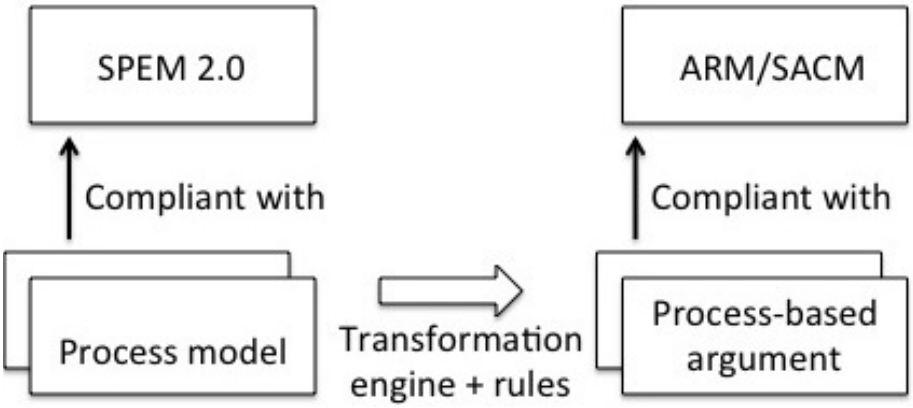

Figure 14: M2M Transformation in MDSafeCer

MDSafeCer transforms models representing single processes into models representing single process-based arguments. THRUST [31] is a method that proposes the transformation of models representing family of processes into models representing facility of process-based arguments.

In the context of SafeCer, Workflow Engine for Analysis, Certification and Test (WEFACT, which is a tool that offers capabilities for modelling workflows and has the potential to integrate external tools) was extended with an additional feature, called MDSafeCer-WEFACT, that implements MDSafeCer principles.

Another related work is presented in [84] by integrating prescriptive elements from the standards into a goal-based safety case. Their approach is based on transformation of tables, presented on the standards, into claims. Prescriptive standards requested that certain analysis, methods, techniques or activities to be performed during the development. These analysis, methods or activities are recommended or highly recommended based on the critical level for the function, and they are grouped in tables. Stensrud proposed to transform these tables into claims, then, the assessment and ratings of the requirements in the standard should be explicitly stated with more detailed arguments.

The OPENCOSS project proposed an automatic argumentation generation based on compliance models of standards. The generation is based on the transformations proposed in [84] and in [89]. In this approach [85], the activities prescribed by standards are transformed into top claims. The references requirements, which those activities should fulfil, are transformed into sub-claims of those top claims. The sub-activities are also transformed into sub-claims and the artefacts are turned into information elements.

Few other approaches targeting model-based certification for process compliance are proposed in literature. The reader may refer to the related work section that is presented in [25].

3.2 Product-based reuse

In this section product-based reuse-oriented works are discussed. We consider the reuse of architectural artefacts and components (out of context → in context). Additionally, we consider the reuse in the domain

safety argumentation overviewing more in general reuse of both product-oriented evidence (classified into immediate and/or direct according to [91]) and product-oriented argumentation artefact approaches.

3.2.1 Product-based patterns of reusable reasoning

The basis for safety case argument reuse lies in the argumentation patterns of recurrent (and thus reusable) reasoning [1]. Just as design patterns, safety case patterns aim at capturing solutions for recurring problems in safety argumentation. Throughout the years many safety case argumentation patters have been proposed [1], [2], [3], [4], [5], [6], [7].

One of the most ubiquitous patterns, in safety argumentation, is the "Hazard Avoidance Pattern" [1]. The pattern implicitly assumes that safety is equivalent to hazards avoidance and that safe means hazards-avoided, hence the top-most goal ("the system is safe") is decomposed by arguing over each identified hazard, assuming that all plausible hazards are identified.
This argument pattern is in line with many safety standards that require each identified plausible hazard to be sufficiently addressed. Instantiating the pattern requires all the different types of evidence [91] as solutions, including immediate, direct and indirect.

Alexander et al. [3] present a collection of safety argumentation patterns for justifying adaptive system safety. Due to the difficulty of arguing safety for adaptive systems simply based on the development process, the work presents fourteen safety case argument patterns that can be used to argue safety of adaptive systems in a product evidence-based manner. The proposed approach for arguing safety for adaptive systems distinguishes between different motivations behind using adaptive systems. For example, an adaptive system can be used to improve or maintain safety with respect to the conventional non-adaptive system. Depending on the motivation, different strategies are used to assure safety.

Similarly, Palin and Habli [5] define a pattern catalogue of ISO 26262-compliant automotive safety arguments together with an overall pattern guidance for connecting the different product and process-related patterns.

Ayoub et al. [6] present a safety case pattern for a model-based development approach, while Bate and Conmy [7] present a set of patterns for a component-based software engineering approach. Wu [4] presents a set of argument patterns and a supporting method for producing architectural safety arguments. Instantiating the higher-level patterns, which start from the top goal ("the system is acceptably safe"), usually require all the different types of evidence as solutions, including immediate, direct and indirect. Some more specific lower-level argument patterns, such as confidence argument pattern for assuring confidence in a piece of evidence [7], usually do not need all the different types of evidence, but can be constrained to a single group, e.g., indirect evidence for the confidence argument pattern.

A group of works that focus on automated generation of safety arguments [8], [9], [10], [11] rely on the notion of argumentation patters. The works build upon the predefined argumentation patterns that are used as the target argumentation models. For example, the works by Armengaud [8] and Basir et al. [9] focus on automating the compilation of the safety case arguments from pre-existing work products.

Denney and Pai [10] focus on automating the assembly of safety cases based on the application of formal reasoning to software. The assembly combines manually created higher-level argument-fragments with automatically generated lower-level argument-fragments derived from formal verification of the implementation against a mathematical specification. The work uses the AutoCert tool [121] for formal verification where the provided specifications represent formalised software requirements.

A work by Prokhorova et al. [11] relies on formal modelling techniques supported by the Event-B formal framework. The work proposes a methodology for formalising the system safety requirements in Event-B

and deriving a corresponding safety case argument from the Event-B specification. The work classifies safety requirements by the way they can be represented in Event-B and proposes a set of classification-based argument patterns to be used for generating specific arguments for each of the requirements classes.

However, one of the most important concepts for reusable reasoning is to justify why the reuse is feasible. In this area we should mention the work of Ruiz [86] where she describes an argument pattern focused on reuse feasibility (see Figure 15). The reuse is analysed based on four pillars:

- Functionality: this aspect is critical for reuse. The main rational for a reuse is to provide the same functionality as needed in the same environment under the same operational conditions or at least compatible conditions.
- Compliance to standard: this aspect refers to standard objectives fulfilment by the reuse, showing compliance with the requirements of the applicable standard, as required in the environment where the component is reused.
- Criticality levels: this aspect is referring to the highest criticality level for which the component was initially developed. In safety-critical systems, functions are associated to hazards and the associated level of severity of these hazards in case of occurrence. Based on this criticality level, the need for requirements coverage might differ. Components should only be reused in environments in which the critical level they are required is the same (or lower than) as the one they were developed for.
- Business case: concerning this aspect, we should look if the IP (intellectual property) rights are not violated or if it is part of the strategy defined by the company or if the reuse is worth the cost of the component.

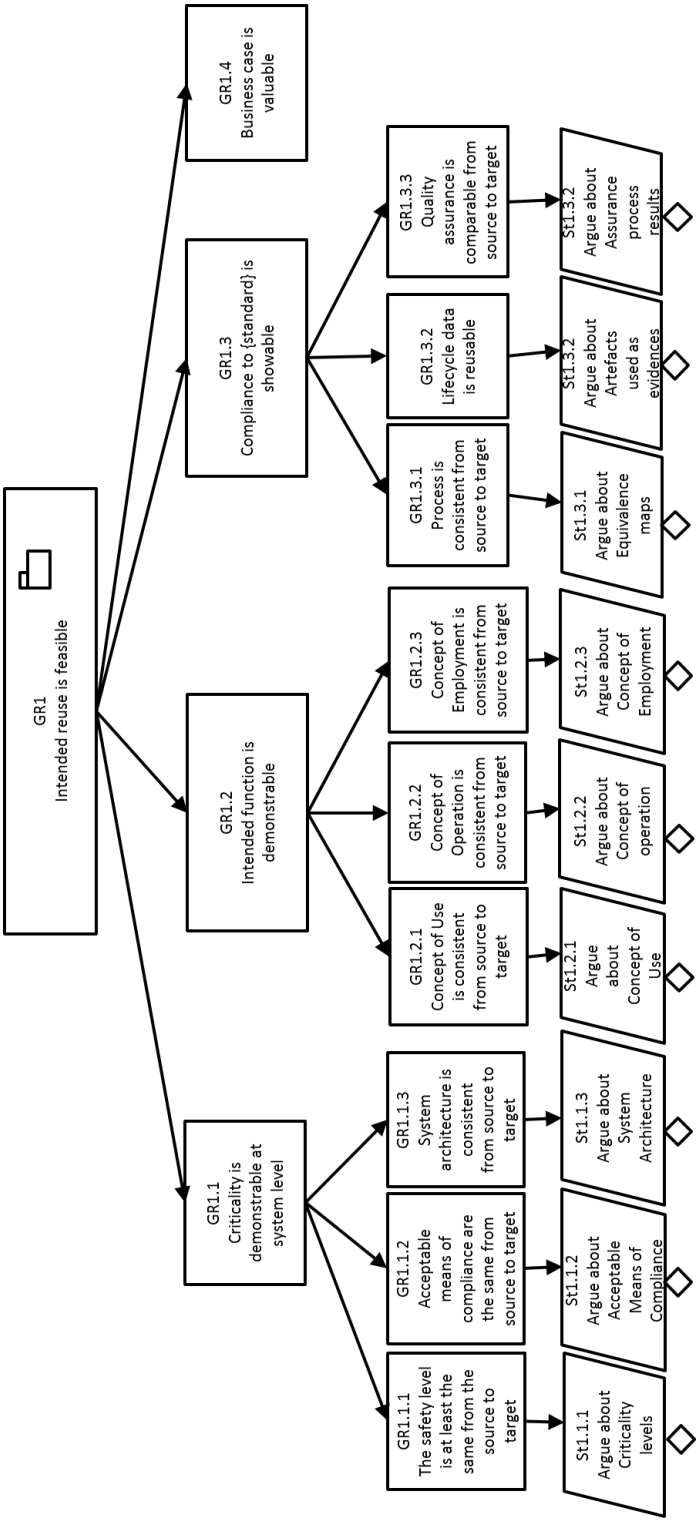

Figure 15: Argumentation pattern for reuse feasibility [86]

Case-based reasoning has also been proposed in [90] to support the representation, retrieval and reuse of assurance case fragments according to certain product characteristics. The approach is supported by the existence of patterns of safety cases. The historical cases are stored and parametrized in order to provide the most suitable assurance case already instantiated to suit the needs of the new context.

3.2.2 Product line engineering

Product Line Engineering (PLE) is often used to facilitate reuse of safety case artefacts including immediate, direct and indirect evidence [12], [13], [14], [15].

Habli and Kelly [12] present an approach to build a product-line safety case so it can be reused for the individual products of the product-line. The authors show how binding the design and argumentation variation points can be used to facilitate reuse of safety case artefacts across a family of products.

Gallina et al. [13] present the VROOM & cC approach that provides means to specify, manage and trace commonalities and variabilities in different parts of the ISO 26262 safety process, in addition to the design phase. The approach considers the impact of different choices within a feature diagram in the corresponding safety case argument.

Reusing safety artefacts requires that variability within them is managed. A PLE-based approach shows how variability can be integrated into the functional safety models by combining functional safety and variability modelling tools [14]. Another approach focuses on Trusted Product Lines by forming a framework for demonstrating that the derived products are fit for purpose in high-integrity civil airspace systems [15]. The work aligns PLE with civil airspace safety standard recommendations on development and integration of reusable elements.

3.2.3 Component -and contract- based engineering

Component-Based Software Engineering (CBSE) is a well-known engineering method for dealing with complex systems. As systems grow in complexity, so does the trend in using components. CBSE is often combined with contract-based design. Traditionally, the concept of contract is defined as a pair of assertions C = ⟨A, G⟩ where a component makes assumptions A on its environment, and if those assumptions are met it offers guarantees G in return. In simple terms, a contract appears as a means for connecting components together thus allowing them to interoperate. More specifically, contracts record agreements in terms of assumptions and guarantees as is described in [SPEEDS D2.5.4]. Assumptions are the assertions that a component assumes are met by other components in the system, and guarantees are the assertions that the component itself is expected to meet in front of other components in the system.
The ultimate goal of contract-based design is to allow for compositional reasoning, stepwise refinement, and a principles-based reuse of components that are already pre-designed or designed independently. Then, one can reuse a component by checking the contract refinement.

The literature ([16], [17]) related to the well-known “3Cs Model” illustrates the three principal issues with the creation and use of reusable components. “3C” stands for: Concept – the abstract semantics of the component, Content – its implementation, and Context – its environment. The third “C” is in general considered the most problematic for safety- and security-critical systems [17]. The notions of Safety Element out-of-Context (SEooC) within the automotive ISO 26262 standard and Reusable Software Components (RSC) within the airborne AC 20-148 standard have been introduced as a way to balance the issue of context and the reuse of components and the corresponding certification. Via these notions SOUP-labelled COTS can get a pedigree, i.e., can be considered as SEooC or RSC under certain conditions.

Fenn et al. [18] present an approach to incremental certification that uses ``informal'' contracts for generation of modular safety case arguments. The approach uses Dependency-Guarantee Relationships (DGRs) that capture the important guaranteed properties of a software component, and define the properties that those guarantees depend on. By relating components with safety-case modules, reuse of components can contribute to reuse of the related modules. This is especially interesting for suppliers of Commercial of the Shelf (COTS) components.

While the use of COTS has many benefits, its use within safety-critical systems requires a systematic approach to their selection, evaluation, and integration. Ye proposes one possible systematic approach in [19]. The approach uses the rely/guarantee reasoning to capture the relationship between COTS and the application, and with that it supports COTS evaluation and selection. The work also specifies a set of argumentation patterns for arguing about the safety of COTS.

Sljivo [20] present an assumption/guarantee contract-based approach to facilitate reuse of certification artefacts together with reusable components such as SEooC within ISO 26262. The contracts in this context represent formal specifications about the guaranteed behaviour of the component and the assumed behaviour of its environment. Besides the potential that assume/guarantee contracts offer with respect to selection and evaluation of COTS/SEooC components, the work maps the component and argumentation meta-models to automatically generated argumentation fragments from the contracts. The argument fragments are generated from pre-defined argumentation patterns. The potential for reuse spans from immediate evidence, to direct and indirect evidence.

Ruiz et al [90] perform a preliminary study towards an approach for enabling compositional safety assurance. In their study, they point out that, in order to support safety assessment, failure behaviours of components, and their behaviour in the presence of failures, must be defined. They recognize that a primary challenge is identifying all of the assumptions made and secondly envisaging all of the different contexts in which the element might be used. Regarding assurance, they also recognize the need to define the set of claims that need to be made concerning a component, in order to support its certification against a particular safety assurance standard. The data required for assurance can be classified in three main categories:

- Artefacts: referring to the data required by an entity when performing the safety assessment.
- Properties: these are characteristics that must be present in the component and after its integration in order to confirm that there are neither concerns, nor an emerging unknown behaviour. The properties need to be verified, and the verification needs to be included as part of the evidence.
- Processes: refers to the activities that shall be performed in order to prepare the reuse and after the reuse itself in order to comply with the standards requirements.

The expert on standards should contribute in defining the integration of the different components and the sharing of responsibilities between the component safety manager and the integrator.
Ruiz et al. point out that the expression “component composition”, entails relationships that exist between concepts from different reference frameworks. These frameworks are related to each of the components that are being integrated.

In managing complex industrial projects, it is a common practice to decompose the project into different subprojects for different development teams. Each of these teams is expected to focus on just one of the subsystems in which the whole project is decomposed. These teams can be from the same company but also be part of different companies, which are suppliers of the manufacturer. Each team should see its development-mission as a project of which they need to assure compliance. By decomposing a project into different subprojects, a hierarchy emerges. A subproject might focus just on the development of a software component or on the hardware of a component. This divide-and-conquer approach can be used to manage the assurance of an overall system.

It is also necessary to open up another project to manage assurance aspects of the integration of components. This integration project should include all the component assurance projects as subprojects. The integration assurance project will include information on the compliance requirements regarding integration and should reference to activities done on the components that are being integrated.

For each component, we should create a dedicated assurance subproject. For each of these subprojects, we need to define which standard(s) and requirements apply to the specified component and up to what level we need to comply with. We also need to specify for each of the subprojects the evidences that we will provide to support assurance of those compliance requirements. In some cases, evidences could be a composition of evidences coming from other subprojects, so we should address that on the project evidence model.

All these approaches can easily be sustained with the meta-models we have described in previous sections. The assurance project meta-model described includes the concept of subprojects. The new integration assurance project should be treated just as another assurance project but with the particularity that the baseline should include all of the post-conditions which are necessary for a complete integration.
For each of the components we need some contract data relating to a component including information about assured properties and behaviours of that component, the artefacts that should be accessible to the authorities and the evidence of the process and activities executed to fulfil the component's assurance requirements. All this information is related to the component assurance information that the integrator should collect during the assurance process of the overall system. Contract refers to the artefacts and their properties, and the rest of the information from the artefact model is considered as a black box.

Finally, contracts can be used for generation of fault trees (CBSA). In that case there are different possibility of reuse as it was developed in the SafeCer project. First of all, it reuses the nominal contract-based specification for performing safety analysis. Moreover, it is a compositional technique and therefore, if a component is composed of subcomponents, then the generation of the fault tree can reuse the fault trees already generated for the subcomponents. Finally, it supports stepwise refinement, so we can generate the fault tree of a high-level architecture, and then we can further decompose and detail some of its components, so we can reuse the previous fault tree by expanding the undeveloped events corresponding to the failure of the components that have been decomposed. These undeveloped events become therefore intermediate events in the new fault tree and they are linked to the failure of the basic components in the new expanded architecture. This refinement can be achieved also by using the fault tree obtained from the model-based analysis of the behavioural model of some components: given a behavioural model of a leaf component and a fault tree generated by fault injection and model-based safety analysis, this fault tree can be attached to the fault tree obtained with CBSA by expanding the undeveloped event corresponding to the failure of the component.

3.2.4 Model-based product certification

Integration of model-based engineering with safety analysis, in order to ease the development of safety cases, is presented in [21]. The work presents how the EAST-ADL2 architecture description language can be used to support the development of safety-critical systems. More specifically, the work integrates the safety case meta-model, the safety analysis, and system modelling. The work promotes reuse in the safety-case domain by binding the safety case with the systematic development, supported by EAST-ADL2.

Habli [22] proposes a model-based assurance approach for facilitating reuse of safety assets within a product-line. Just as the product-line reference architecture represents the base for deriving product architectures, the product-line safety case can play the same role for deriving arguments for explaining why the particular product is acceptably safe to operate in the particular environment. The proposed approach for the product-line safety case development extends the argumentation notation to include product-line elements to handle variabilities within the argument. By capturing the variabilities and the underlying context assumptions, the approach can be used to reuse safety assets together with the used product-line assets.

Hawkins et al. [23] proposes a model-based approach for automatic generation of assurance cases from automatically extracted information from the system design, analysis and development models. The

approach ensures traceability and consistency between the reference information models and the generated assurance arguments. The proposed approach uses “model weaving” to capture the dependencies between the reference information models and the assurance argument patterns. The Model Based Assurance Case (MBAC) program is at the core of the prototype tool that implements the approach [24]. MBAC takes the argument pattern, reference information, and weaving models as its input together with the corresponding meta-models, and provides an instantiated argument model as the output.

3.3 Cross-concern reuse

In this section, we discuss cross-concern reuse. Safety and security represent two concerns, which have been in focus within two distinct communities. The relevance of harmonizing and cross-fertilizing the safety and security communities is well known. Within the MAFTIA project [72], researchers have worked on a common terminological framework.

The need of combining safety and security- related certification processes, in order to save time and money as well as complexity via reduction of duplications, has been recognised for more than a decade. Within the SafSec project [73] a common methodology for security accreditation and safety assurance was developed [71], [74]. This methodology is based on recognizing that both safety and security processes recommend risk-driven development processes. Safety hazards and security threats are both analysed via a unified risk management process. The approach, known as dependability by contract [75], aims at developing a unique dependability case, incorporating both security and safety concerns via a compositional contract-based approach.

More recently, to enable cross-concern reuse of process elements, the notion of a Security-informed Safety-Oriented Process Line is introduced [34]. This extends the notion of “Safety-oriented Process Line” towards reuse of process-related information in the context of multi-concerns. For sake of clarity, it should be noticed that the modelling capabilities that are explored in order to model Security-informed Safety-Oriented Process Lines and more specifically the cross-concern reuse are not mature yet.

4. State of the practice

In this chapter, an overview of the state of practice with respect to cross-and-intra domain reuse is given. The overview is given per domain. For each domain, first of all an introductory paragraph shortly indicates the requirements and recommendation coming from the related standards in terms of reuse, then the topics that were considered in Section 2 are also considered here, when appropriate. Additionally, an overview of cross- domain practices is also given.

4.1 Automotive domain

ISO 26262:2011 [108], the standard for functional safety within the automotive domain, mentions reuse explicitly in some of its parts. These parts are:

- Part 3 together with Part 8 (change management), indicate what should be done in case of a modification of a pre-existing system in order to determine the impact on the system-related work product as well as in terms of process-related documentation (how to tailor the life cycle).
- Part 10 (Section 9) introduces the notion of the Safety Element Out of Context (SEooC) and recommends corresponding processes for software and hardware elements' developments and integration. These processes provide guidance on how to reuse components developed out of context.
- Part 8 (Sections 12, 13) respectively recommend how to qualify pre-existing software and hardware components. This part (Section 14) also introduces the notion of proven in use argument to achieve compliance with ISO 26262 when field data is available for an existing and expected component to be reused.

It should be noted that ISO 26262-2011 underwent a revision period and a new version is about to appear. Currently, the new version is still in draft status ISO/DIS 26262 [125]. Despite the revision, in case the draft version is approved, the above-given information will remain valid.

4.1.1 Process-based reuse

In general, we see a two-step approach in the automotive domain for the process-related reuse activities, namely the company-specific process and project-specific process.

- The general company specific process can be seen as a process framework, which defines specific tools, document templates for work products, provide guidance including best practices for specific activities and defines roles and responsibilities,
- The project specific process is tailored for the project specific needs and it is based on the company specific process.

At the moment, there is no systematic reuse of process elements performed in the automotive industry to derive these project specific processes. More ad-hoc reuse techniques such as clone-and-own [122] are used here, without sophisticated and systematic process improvement strategies.

The processes are typically modelled by using semiformal notations: Business Process Model and Notation (BPMN) or SPEM 2.0.

4.1.2 Product-based reuse

In the automotive domain, three important reuse techniques for product-based reuse are:

- **Design Patterns** – a way of reusing abstract knowledge about a design problem and its solution. A pattern is a description of the problem and the essence of its solution.

- **Application frameworks** – Frameworks are a sub-system design made up of a collection of abstract and concrete classes and the interfaces between them. Frameworks represent moderately large entities that can be reused.
- **Application system reuse** (Software Product-Lines, COTS) – Involves the reuse of entire application systems either by configuring a system for an environment or by integrating two or more systems to create a new application.

Within the engineering life-cycle phase, reuse is typically (especially according to the experience of KPIT medini Technologies), conducted at:

1. Software Level (e.g. Software Libraries, AUTOSAR compliant SW COTS; Software Platforms).
2. Hardware Level (e.g. Hardware COTS such as sensors, ECUS, actuators; Hardware Platforms).
3. System Level (e.g. Standardized systems such as HV batteries, electric motors, ABS, ESP, braking).

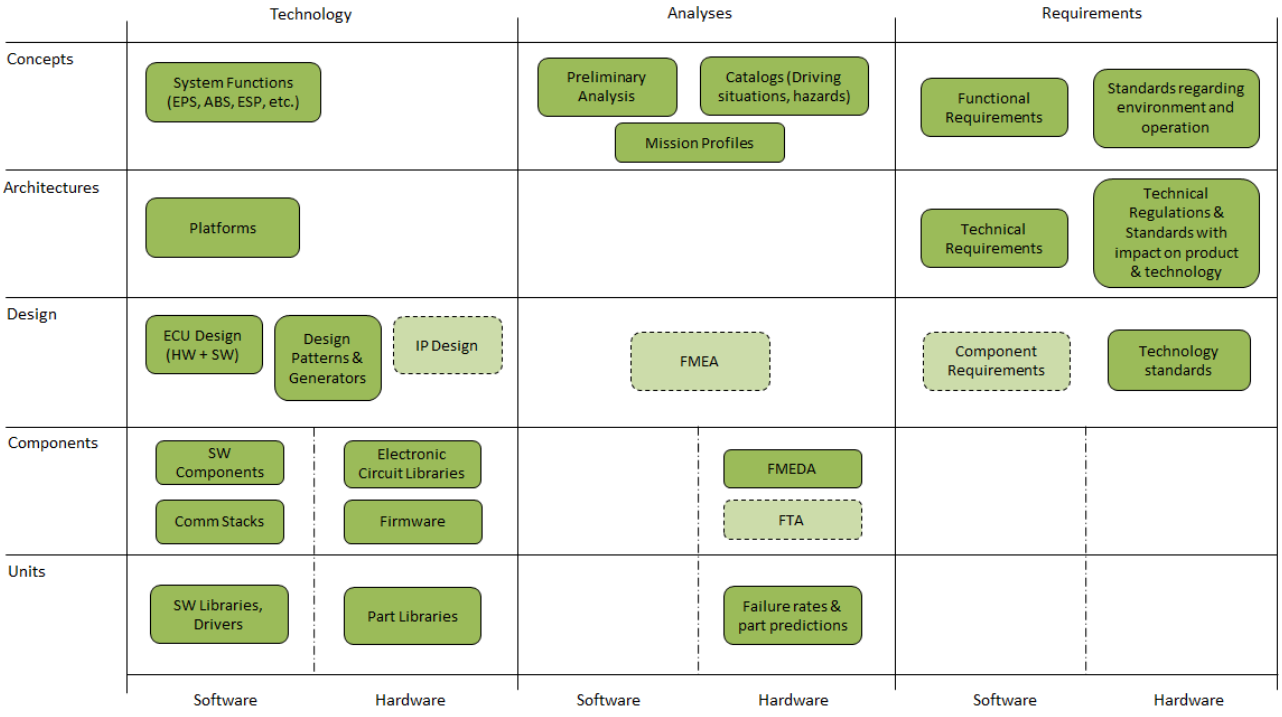

Figure 16: Areas of reuse along the development lifecycle (green: typical reuse, light-green w/dashed line: limited reuse)

Within today's automotive product development lifecycle, reuse can be observed in a number of areas and for various concepts, artefacts, designs, requirements, or components. Figure 16 depicts a summary of the main areas and typical artefacts that are subject of reuse. This summary is based on the experience of KPIT medini Technologies. These areas are loosely oriented along the V-model and process hierarchy of ISO 26262 and Automotive SPICE for system development, in detail:

- *Concept phase*: OEMs and suppliers apply reuse already in early phases of concept development, for example if whole vehicle functions and or subsystems are taken over from one project/product line to another. Besides the pure concepts, the reuse realizes in taking over complete sets of requirements from previous products, sometimes with slight adaptations and/or extensions. Especially at the company level, standards are maintained regarding environment and operation of the vehicle (e.g. homologation/decommissioning acc. to FAA/EU directives, etc.). For analyses, safety engineers reuse parts of preliminary analyses of similar systems as base for a new product. For example, the hazard analysis and risk assessment is usually based on common catalogues for driving situations, mission profiles (collected per vehicle/system type), or "master documents" that

are adapted for a specific product. Hence, altogether this includes preliminary ASIL ratings and safety goals that represent the base of a new development.

- *Architectures*: At system level, many companies maintain some sort of *reference platform* in order not to reinvent the whole system architecture every time, and to benefit from reusing components. Examples include electrical architectures at OEMS (e.g. Volkswagen with the MQB, General Motors with the "Global A/B" platforms), "base ECUs"/platforms at various tier 1 suppliers (e.g. variants of a certain ESP ECU for different vehicle types) or product lines for chips at semiconductor companies (e.g. microcontroller families such as the AURIX series from Infineon). These system architectures are often accompanied with constraints expressed as requirements from technical regulations and standards (e.g. ECE series of standards, IEEE standards, etc.). From a safety analysis point of view, reuse of safety analysis at the architecture level is seldom.
- *Design*: Reuse at the detailed system design including hardware, software is similarly organized, and the transition from architecture to design is fluent. For example, a specific ECU design including safety mechanism and shutoff paths for actuators is subject of reuse (e.g. 3-level-monitoring) and typical built-in features are carried over from one product to another (e.g. voltage monitoring, watchdogs, etc.). At the same time the technology or main components are reused (types of sensors, HW components such as micros, H-bridges and drivers, SW design and libraries), etc. For the semiconductor companies IP designs are typically subject of reuse across several products, e.g. an ALU or memory specification in VHDL. However, reuse is mainly achieved via a "clone-and-own" of work products, so that a formalized reuse approach, by product lines or similar, does not exist. The same is especially true for the corresponding safety analyses. For example, teams often copy FMEAs from the previous products with a subsequent adaptation to the new design.
- *Components*: Hardware and software components are reused frequently in the automotive domain, simply because they normally provide a self-contained functionality. While for HW the reuse is often simpler since its electrical interface is usually stable, for SW the adaptation into the new product context usually requires at least configuration (e.g. AUTOSAR SW components, RTE, communication stack, and MCAL configurations, etc.), or patches (e.g. firmware updates for changed HW). This does not necessarily mean a 100% take over, but many HW/SW components are reused in new products. Examples for a direct carry over include sensors (e.g. wheel speed or hall sensors), microcontrollers, voltage regulators, H-bridges and gate drivers, motors, etc., although the circuit might be adapted due to housing changes or changes of the outside electrical interface/connector. Some companies maintain building block libraries of circuits so that "logical components" can be reused that implement a certain function. With respect to safety analyses, the FMEDA, and sometimes FTAs too, are copied as basis for a new analysis.
- *Units*: At the lowest level, engineers reuse units such as SW libraries or functions and HW parts, managed in libraries. These parts constitute the individual building blocks and reuse is rather ad-hoc without specific planning. On the analysis side, safety analysts maintain libraries for failure rates and hardware failure predictions in order to adapt them to a new product, based on its mission profile.

4.1.3 Cross-concern reuse

No specific approaches for safety and security concerning cross concern reuse are utilized in the automotive domain. The scope of the current version ISO 26262:2011 (V1) is only on safety-critical aspects of car passengers and pedestrians, therefore the cross-concern reuse is not in the focus of the automotive domain. No security-related aspects are covered in the ISO 26262 yet.

The upcoming revision of the functional safety standard ISO 26262:2018 (V2) will extend the scope from car passenger to truck, bus and motorbike passengers, which will lead to specific differences, commonalities and optionality's for these domains. Furthermore, the relationship between safety and security concerns

are also addressed, particularly by proposing an interface between Safety and Cybersecurity teams, and adding one mandatory requirement (similar to IEC 61508 (2010)) to consider the safety impact of security threats during the life cycle. This was initiated by AMASS partners who have been members of the task group tackling this issue. However, the security subject in the Automotive has somehow been recently anticipated by SAE J3061 – "Cybersecurity Guidebook for Cyber-Physical Vehicle Systems", as of Feb 2016 [122]. Its authors, in consideration that cybersecurity and functional safety share parallel processes (e.g. threat analysis and risk assessment vs hazard analysis and risk assessment; attack tree analysis vs fault tree analysis), and its teams need to interact, have adopted the approach of tailoring the cybersecurity process from "ISO 26262: Road vehicles – Functional safety." Therefore, the new ISO 26262 version, expected to cover also cyber-security aspects, might likely limit to refer said Guidance, with no repetitions. In the mean-time (2016), two New Work Item Proposals have been accepted by ISO TC22 concerning Cybersecurity in the Automotive Domain, which will result in one standard and Working Group:

- **"Road vehicles – Automotive Security Engineering"** (German DVA and DIN)
- **"Road vehicles – Vehicle Cybersecurity Engineering",** SAE, based on the existing SAE Guideline J3061 (Cybersecurity Guidebook for Cyber-Physical Vehicle Systems).

The aforementioned automotive cyber-security standard SAE J3061 highlights the interaction between security and safety but reuse aspects are not covered in this standard.

Besides SAE J3061, a new standard, ISO/SAE AWI 21434 [126], for road vehicles' cybersecurity is under development and expected to be published in 2019. AMASS members are involved in the development of this standard. At the time being, however, its status is still confidential.

It is however possible to mention that within ISO/SAE AWI 21434, safety is considered as one of the properties, which needs to be protected by cybersecurity. Other properties are confidentiality of sensitive data, financial impact or availability.

Safety engineering, however, is considered as a related discipline. Based on this there are multiple interactions between safety and security expected. In addition, the product development process for ISO/SAE AWI 21434 is following a similar structure than the product development for ISO 26262. This was done because ISO 26262 is not only used for safety engineering but is also describing the general system engineering approach used in the automotive domain. Following a similar process should support the adoption (process structure and phases are easily understandable for all automotive experts) and it should support the co-engineering/cross-concern reuse (phases can be combined since both standards have the same phases).

Given the development of a safety as well as security-critical vehicle system, it is assumed that the first input comes from safety since safety can determine hazards already based on expected functions. As soon as a first refinement, e.g. a first system concept is available, safety and security can be analysed and safety and security goals can be defined. It is expected that there are safety-oriented security goals which are identified in cooperation between safety and security. Goals are translated to requirements and further towards implementation. At the time being, however, it is still matter of discussion how far safety and security interact. It is certain that requirements and implementation need to be coordinated between safety and security. It is unclear if the standard should push towards combined safety & security measures because there is a huge discussion regarding the maintainability of such measures. There is also an ongoing discussion regarding responsibilities and placement of assurance. This relates mainly to the issue if safety related security requirements and how their assurance should be contained in the safety case.

4.2 Aerospace domain

The aerospace industry has a very strong and highly regulated safety culture. For each development level (i.e., system, subsystem) a safety regulation is available. ARP4754A [106], for instance, focuses on the system development; while DO-178C/ED-12C [104] and DO-254/ED-80 [110] provide guidance concerning

software and hardware development respectively. In addition to the guidance related to the development, other guidance is available to address relevant concerns. ARP4761 [107], for instance, focuses on safety assessment.

Moreover, the Manufacturer usually utilizes own process standards and supplementary documents which normally are more stringent than those required by the certification authority.

Concerning software reuse, for instance, DO-178C Chapter 12 'Additional Considerations' deals with reuse. The subchapters 'Use of Previously Developed Software', 'Modifications to Previously Developed Software', 'Change of Aircraft Installation', 'Change of Application or Development Environment' and 'Upgrading A Development Baseline' provide guidance for such cases. These sub-chapters are also complemented by the Advisory Circular AC 20-148 "Reusable software components". Although dated 2004, and thus still referring DO-178B, its concepts apply also to DO-178C.

Given the interplay of system-level and subsystem-level standards, when reusing software, not only the related chapters in the DO-178C have to be considered, but also the related chapters in the ARP4754 specifically chapter 11 'Modified Aircraft' in the ARP4754. The chapters explain the impact on the whole system development as well as the safety assessment process.

In the context of ground-based systems (e.g., Air Traffic Management-related software), DO-278/ED-109 [111] applies. Before the introduction of DO-278/ED-109, application of DO-178B/ED-12B [105] was requested. Due to the extensive use of COTS software within ground software, a different guidance was needed. DO-278/ED-109 borrows from DO-178B/ED-12B, and a large part of the document explicitly references DO-178B/ED-12B. Latest edition DO-278A/ED-109A fully mirrors DO-178C/ED-12C, and also makes use of its same four supplements. Those supplements are covering model-based, object-orientation, formal methods, and tool qualification topics, as is done by similar complements of DO178C, which indicate the additions, modifications, and deletions to DO-178C/ED-12C or DO-278A/ED- 109A objectives when a specific technique or method is used as part of a software life cycle, and provide any additional guidance required.

In the cybersecurity area, the relevant standards, which have been developed, are:

- RTCA DO 326A/ED202A "Airworthiness Security Process Specification", one of the first standards covering the integration of safety engineering and security engineering.
- RTCA DO 355/ED204, "Information Security Guidance for Continuing Airworthiness" covers operations and maintenance".
- DO-356 "Airworthiness Security Methods and Considerations", covers methods and considerations to support DO326A.

4.2.1 Process-based reuse

Within aerospace standards there are no explicit recommendation for process reuse. However, as mentioned before, in the context of the DO-178C and DO-278A/ED- 109A families, for instance, supplements are provided as well as circulars that guide how to use the baseline in combination with the supplements. Even if not explicitly mentioned, variation points in the baseline are indicated in terms of additions, deletions, and modifications.

Within aerospace industry, however, these still implicit variation points are not exploited. Typically, based on the baseline, generic processes are modelled and then the needed variations are performed manually. In this respect, process-based software reuse is somehow faced in the aforementioned FAA Advisory Circular 20-148 "Reusable software components".

4.2.2 Product-based reuse

Aerospace industry typically adopts the following approaches in order to reuse product-related artifacts:

- The first approach in terms of reuse is the definition of Commercial of-the-shelf (COTS) as a part of so-called System-on-Chip (SoC) devices. The goal is to increase the production rate and production volume, and to significantly reduce the production cost significantly for aviation applications. It is a trend to observe that the aviation industry is trying to adopt this technology at a high scale integration level also in airborne critical applications. For the development have to be applied DO-178C for software or DO-254 for hardware.
- The Integrated Modular Avionics (IMA) [112] is the second approach for reuse technologies. The goal is to build a flexible hardware and software platform with core functionality that can be easily extended and customised to meet specific requirements. The IMA could be used on different aircraft models with the same platform that provides services, as a shared set of flexible, reusable, and interoperable hardware and software components, designed and verified to meet a defined set of safety and performance requirements, in order to host applications performing aircraft functions. The IMA architecture integrates many aircraft functions in the same platform. Those functions are provided by several hosted applications that have historically been contained in functionally and physically separated 'boxes' or 'line replaceable unit' (LRUs). IMA platforms are composed of modules which are designed to be reusable in order to reduce development costs and occasionally facilitate certification programs. Some modules provide only mechanical functions, such as cooling and electrical power supply, others include core software and associated computing capabilities.
- Another way to allow reuse is the usage of FPGA. They represent a good example of a reconfigurable component used in digital systems. Reconfigurable devices with SoC buses and reusable IP are building blocks that can be used to build larger modules that can be reconfigured to perform multiple system functions and thus they are reusable in multiple platforms.

Each characterization of "reusable" modules and functions, their associated attributes, their configurability and their performances have to be verified commensurately according to the EUROCAE/SAE document ED-79A/ARP 4754A 'Guidelines for development of civil Aircraft and Systems' and their respective Development/Design Assurance Level. After that, the failure conditions at equipment level have to be classified because they may change as a result of a particular aircraft installation architecture, characteristics or mitigation. The document has to be used as well as a guidance to ensure a proper development, validation and verification of the ETSO (European Technical Standard Order) and the functional equipment requirements according to the standards.

4.3 Industrial automation domain

Within AMASS, the case study related to Industrial automation is expected to make use of the meta-standard IEC-61508. Indeed, the IEC 61508 standard series is the reference for generic industrial automation, and it represents the "parent" of more specialized standards, as outlined in Figure 17.

Figure 17: IEC 61508 as the meta-standard for functional safety, and its derived domain-specific standards

Thus, in this section, we recall the requirements on reuse pertaining to that standard. First, the standard acknowledges the "proven in use" argument as possible justification for the utilization of an existing element (either integrated system, or hardware or software):

(IEC 61508-2, §7.4.10.1): "*An element shall only be regarded as proven in use when it has a clearly restricted and specified functionality and when there is adequate documentary evidence to demonstrate that the likelihood of any dangerous systematic faults is low enough that the required safety integrity levels of the safety functions that use the element is achieved*".

Once a proven in use element is subject to modification, as it is usually the case (the reuse-as-is is not systematically feasible), requirements in IEC 61508-2, §7.8 (*E/E/PE system modification*) apply. Among these:

7.8.2.1 Appropriate documentation shall be established and maintained for each E/E/PE (Electrical/Electronic/Programmable Electronic) system modification activity. The documentation shall include:

a) the detailed specification of the modification or change;

b) an analysis of the impact of the modification activity on the overall system, including hardware, software (see IEC 61508-3), human interaction and the environment and possible interactions;

c) all approvals for changes;

d) progress of changes;

e) test cases for subsystems and elements including revalidation data;

f) E/E/PE system configuration management history;

g) deviation from normal operations and conditions;

h) necessary changes to system procedures;

i) necessary changes to documentation.

On that basis, process and product requirements are derived as described below.

IEC 61508-3 referred with respect to "proven in use" software (route 2s) to Part 2 and the above-mentioned paragraph, which is by declaration not valid for software. This was a mistake made during the editing process, so an IEC TS 61508-3-1 "Part 3-1: Software requirements - Reuse of pre-existing software elements to implement all or part of a safety function" was created to specify more rigid the requirements to accept software as "proven-in-use" (2016) (see 5.3.2).

4.3.1 Process-based reuse

As mentioned before, in compliance with *7.8.2.1* appropriate documentation shall be established and maintained. The documentation to be established and maintained also includes the one related to the process. The process initially used to develop the element, proven in use, may also require changes.

4.3.2 Product-based reuse

In addition to aforementioned (IEC 61508-2, §7.4.10.1), other process-oriented requirements apply (§7.4.10.x) aimed at guiding the reuse of product-related artefacts. Focus is on collecting the evidence that the (existing) element is meant to be reused in the same (or very similar) conditions as it was initially (and successfully) utilized, in terms of operational profile, factors triggering faults, interfaces, operating system, and human factors (if any). In case of difference, an impact analysis has to be carried out, in order to demonstrate that the likelihood of any dangerous systematic faults is low enough that the required safety integrity level(s) of the safety function(s) that use the element is achieved. At the end, a safety justification shall be documented, confirming that the element supports the required safety function with the required systematic safety integrity. This has to be typically compiled into the **Safety Manual** of the reused (proven in use) element, as ordinarily required for any element that is claimed to be compliant with the standard itself.

In particular, when the existing element is software and this is meant to be utilized for implementing part or all of a safety function, IEC 61508-3 (§7.4.2.12) specifies the three alternative "compliance routes" to be followed for confirming compliance:

- *Route 1S: compliant development. Compliance with the requirements of this standard for the avoidance and control of systematic faults in software;*
- *Route 2S: proven in use. Provide evidence that the element is proven in use. See 7.4.10 of IEC 61508-2;*
  Note: This was superseded by IEC 61508-3-1 to correct the reference to the purely hardware oriented Part 2, which is not sufficient for software.
- *Route 3S: assessment of non-compliant development. Compliance with 7.4.2.13.*

Route 1S simply represents the (lucky) circumstance that the existing software had been developed compliant with the same IEC 61508-3 standard, and for the same integrity level. Route 2S recalls the former proven in use concept which had to be corrected by IEC TS 61508-3-1 (see above). More interesting, from a process point of view, is Route 3S. It refers to the case of cross-domain reuse: existing elements having developed as compliant with other standards (if any). The route refers §7.4.2.13, with its inner requirements:

a) The software safety requirements specification for the element in its new application shall be documented to the same degree of precision as would be required by this standard for any safety related element of the same systematic capability. The software safety requirements specification shall cover the functional and safety behaviour as applicable to the element in its new application and as specified in 7.2. See Table A.1.

b) The justification for use of a software element shall provide evidence that the desirable safety properties specified in the referenced sub-clauses (i.e. 7.2.2, 7.4.3, 7.4.4, 7.4.5, 7.4.6, 7.4.7, 7.5.2, 7.7.2, 7.8.2, 7.9.2, and Clause 8) have been considered, taking account of the guidance in Annex C.

c) The element's design shall be documented to a degree of precision, sufficient to provide evidence of compliance with the requirement specification and the required systematic capability. See 7.4.3, 7.4.5 and 7.4.6, and Tables A.2 and A.4 of Annex A.

d) The evidence required in 7.4.2.13 a) and 7.4.2.13 b) shall cover the software's integration with the hardware. See 7.5 and Table A.6 of Annex A.

e) *There shall be evidence that the element has been subject to verification and validation using a systematic approach with documented testing and review of all parts of the element's design and code. See 7.4.7, 7.4.8, 7.5, 7.7, and 7.9 and Tables A.5 to A.7 and A.9 of Annex A as well as related tables in Annex B.*
NOTE 1 Positive operational experience may be used to satisfy black-box and probabilistic testing requirements [see Tables A.7 and B.3].

f) *Where the software element provides functions which are not required in the safety related system, then evidence shall be provided that the unwanted functions will not prevent the E/E/PE system from meeting its safety requirements.*
NOTE 2 Ways to meet this requirement include:
- *removing the functions from the build;*
- *disabling the functions;*
- *appropriate system architecture (e.g. partitioning, wrappers, diversity, checking the credibility of outputs);*
- *extensive testing.*

g) *There shall be evidence that all credible failure mechanisms of the software element have been identified and that appropriate mitigation measures have been implemented.*
NOTE 3 Appropriate mitigation measures include:
- *appropriate system architecture (e.g. partitioning, wrappers, diversity, credibility of checking of outputs);*
- *exception handling.*

h) *The planning for use of the element shall identify the configuration of the software element, the software and hardware run-time environment and if necessary the configuration of the compilation / linking system.*

i) *The justification for use of the element shall be valid for only those applications which respect the assumptions made in the compliant item safety manual for the element (see Annex D of IEC 61508-2 and Annex D).*

Apart of the required Safety Manual, as usual for any other element, it's worth highlighting that the process simply asks for a substantial coverage (mapping) of the requirements in the standard, as evidenced by the available project-specific artefacts.

Based on experiences in the Swedish context, to understand the evolution towards reuse for functional safety, it is useful to compare with the reuse within the domain of high-availability systems (defined as systems with uptime having five nines 99,999% on a yearly basis). Within this domain (particularly in telecom area), initiatives such as OpenSAF (Ref. www.opensaf.org) and OPNFV (Ref. www.opnfv.org) for open specifications and open source have been promoted. In 2008, to ease the architecture evolution, a reference architecture was released via SCOPE-alliance organization.

The trend that can be witnessed within high-availability systems is getting momentum also within safety-critical systems, and platforms are being implemented.

4.3.3 Cross-concern reuse

Given the rather recent evolution in terms of connectivity within the industrial automation domain and consequent provision of standards that try to cover security aspects, no specific cross-concern and at the same time cross-domain practices seem to be existing. On the other hand, IEC 61508 is the generic standard for functional safety of E/E/PE systems, and has to be applied in any application that is not covered by a related (derived) domain specific standard. It should be noted, that already IEC 61508 (2010) (Ed. 2.0) had requirements and guidance how to consider security issues in case of safety impact. It was the

first functional safety standard taking up this joint safety/security aspect (triggered by today's AMASS partners active in this committee).

4.4 Railway domain

Based on the background section presented in [26], in this section, first of all, the requirements and recommendations coming from CENELEC EN 5012x are briefly recalled. The purpose of this background information is to show that a process engineer, in charge of defining a process, is in reality in front of an implicit family of processes. To provide the project specific process, the process engineer has to select and compose the variants together with the commonality. The presentation is limited to a subset of CENELEC EN 5012x. However, the observation is valid in general for all standards/normative documents containing process related recommendations.

The CENELEC EN 5012x (x: 6, 8, 9) is a family of standards that contains requirements and recommendations concerning processes to be followed for the development and assurance of safety-critical systems. This family of standards is used for the certification of railway systems and signalling control-command equipment. As it was documented within the deliverable D6.1 of the MODSafe project [70], Light Rail, Metros, Trams are still characterized by a diversified landscape of safety requirements, safety models, roles and responsibilities, safety approval, acceptance and certification schemes. However, convergence towards the CENELEC standard series is evident. In this section, we briefly present portions of EN 50126, EN 50129 and EN 50128 in order to let emerge their family-oriented nature.

EN 50126 [101] defines a fourteen-phase process to manage Reliability, Availability, Maintainability and Safety at system level. The Risk Analysis Phase is the third phase. The objective of this phase is multi-fold: 1) identification of the hazards associated with the system; 2) estimation of the risk associated with the hazards; 3) development of a process for risk management. One of the outcomes of the Risk Analysis phase is the assignment of a SIL to any safety relevant function or system or sub-system or component. A SIL specifies a target level of risk reduction and is typically defined in components that operate in a safety-critical system. There are four discrete integrity levels associated with SIL, with SIL 4 being the most dependable and SIL 1 the least. The SIL allocation is made taking into account the rate of dangerous failures and tolerable hazard rate of the function, system, sub-system or component. The SIL of a system to be developed is determined at system level. The software "inherits" the SIL as any other part of the system through decomposition. Then, EN 50128 defines what must be done to develop SW functions with that SIL.

EN 50129 [102] defines the conditions that shall be satisfied in order that a safety-related electronic railway system/sub-system/equipment can be accepted as adequately safe for its intended application. These conditions are constituted of three types of evidence: Evidence of quality management, Evidence of safety management, and Evidence of functional and technical safety. The documentary evidence that these conditions have been satisfied shall be included in a structured safety justification document, known as the safety case. The safety case shall be structured in six parts. In this sub subsection we limit our attention to the following parts:

- Part 2 Quality Management Report, this shall contain the evidence of quality management, e.g., evidence of adequate organizational structures as well as evidence of adequate personnel competence and training;
- Part 3 Safety Management Report, this shall contain the evidence of safety management, e.g., evidence that the safety management process consists of a number of phases and activities, which are linked to form the safety life-cycle in compliance with EN 50126 and with EN 50128 at software sub-system level. The software architecture design phase should for instance be aligned with the system architecture design.
- Part 6 Conclusion, this shall summarize the evidence presented in the previous parts of the safety

case, and argue that the relevant system/sub- system/equipment is adequately safe, subject to compliance with the specified application conditions.

It should be noted that the depth of the evidence presented and the extent of the supporting documentation should be appropriate to the SIL of the system/sub-system/equipment under scrutiny.

EN50128 [103] focuses on processes for the development, deployment and maintenance of safety-related software for railway control and protection applications. EN 50128 does not mandate the use of a particular software development lifecycle. It only provides normative tables and recommendations concerning specific process elements, e.g., roles, work products, techniques, tools, tasks. Illustrative software route maps are indicated, however, a process engineer is responsible for the selection and composition of adequate process elements aimed at achieving the required software integrity level.

To give an idea of these normative tables and recommendations, the process elements related to the Software Architecture & Design Phase are recalled.

Tasks and related work products- The design task should receive as input the Software Requirements Specification and should deliver as output the Software Architecture Specification, the Software Design Specification, the Software Interface Specifications, the Software Integration Test Specification, the Soft- ware/Hardware Integration Test Specification, and the Software Architecture and Design Verification Report. The verification task should receive as input all necessary system, hardware and software documentation and should deliver as output a Software Verification Plan, a set of Software Verification Report(s), and a Software Quality Assurance Verification Report. The validation task should receive as input all necessary system, hardware and software documentation and should deliver as output a Software Validation Plan, a Software Validation Re- port and a Software Validation Verification Report.

Guideline- We limit our attention to Annex A. According to Table A.4 (reproduced in Figure 18), formal methods are recommended (R) for SIL 1 and SIL 2 and highly recommended (HR) for SIL 3 and SIL 4, and those recommendation levels are exactly mirrored in Table A.5, formal proofs. More generally, modelling is HR for SIL1-4. According to Table A.17, petri nets are R for SIL 1 and SIL 2 and HR for SIL 3 and SIL 4. Finally, according to Table A.22, Object Oriented Detailed Design is R for SIL 1 and SIL 2 and HR for SIL 3 and SIL 4.

TECHNIQUE/MEASURE	Ref	SIL 0	SIL 1	SIL 2	SIL 3	SIL 4
1. Formal Methods	D.28	-	R	R	HR	HR
2. Modelling	Table A.17	R	HR	HR	HR	HR
3. Structured methodology	D.52	R	HR	HR	HR	HR
4. Modular Approach	D.38	HR	M	M	M	M
5. Components	Table A.20	HR	HR	HR	HR	HR
6. Design and Coding Standards	Table A.12	HR	HR	HR	M	M
7. Analysable Programs	D.2	HR	HR	HR	HR	HR
8. Strongly Typed Programming Language	D.49	R	HR	HR	HR	HR
9. Structured Programming	D.53	R	HR	HR	HR	HR
10. Programming Language	Table A.15	R	HR	HR	HR	HR
11. Language Subset	D.35	-	-	-	HR	HR
12. Object Oriented Programming	Table A.22 D.57	R	R	R	R	R
13. Procedural programming	D.60	R	HR	HR	HR	HR
14. Metaprogramming	D.59	R	R	R	R	R

Requirements:

1) An approved combination of techniques for Software Safety Integrity Levels 3 and 4 is 4, 5, 6, 8 and one from 1 or 2.
2) An approved combination of techniques for Software Safety Integrity Levels 1 and 2 is 3, 4, 5, 6 and one from 8, 9 or 10.
3) Metaprogramming shall be restricted to the production of the code of the software source before compilation.

Figure 18: Reproduction of Table 4, taken from EN 50128

Roles - We limit our attention to Annex B:

- According to Table B.2, a designer shall: transform specified software requirements into acceptable solutions; own the architecture and downstream solutions; define or select the design methods and supporting tools; apply appropriate design principles and standards; develop component specifications where appropriate; maintain traceability to and from the specified software requirements; develop and maintain the design documentation; ensure design documents are under change and configuration control.
- With respect to expected competencies, a designer shall be competent in: engineering appropriate to the application area, the safety design principles, design analysis & design test methodologies, and understanding the problem domain.
- Moreover, a designer shall understand: the constraints imposed by the hardware platform, the operating system and the interfacing systems and the relevant parts of EN 50128.
- Finally, (s)he shall be able to work within design constraints in a given environment.

According to Table B.5, a verifier shall be: competent requirements engineering and experienced in the

applications domain and in the safety attributes of the applications domain. Moreover, a verifier shall understand: the overall role of the system and the environment of application; analytical techniques and outcomes; the applicable regulations; and the requirements of EN 50128.

Finally, according to Table B.7, a validator shall be competent in: the domain where validation is carried out as well as various validation approaches / methodologies and be able to identify the most suitable method or combination of methods in a given context. Moreover, he/she shall be: experienced in safety attributes of applications domain; capable of deriving the types of validation evidence required from given specifications bearing in mind the intended application as well as of combining different sources and types of evidence and synthesize an overall view about fitness for purpose or constraints and limitations of the application. A validator shall also have analytical thinking ability and good observation skills as well as overall software understanding and perspective including understanding the application environment. Finally, he/she shall understand the requirements of EN 50128. It should be also mentioned that the verifier and validator can be the same person in case of SIL1 and SIL2.

4.4.1 Process-based reuse

In general, no systematic method is used in industry to reuse process-related information. In most of the cases, processes are described in natural languages. In some cases, graphical representations obtained by using non-standardized proprietary tools exist. Given the rigidity of such tools, easy-configuration is not possible and variations are specified via notes.

Despite the rather evident but still implicit family or processes that emerges by considering the baseline together with the variation points indicated via the numerous tables within EN 5012x, currently process-based reuse is not conducted by exploiting process line engineering approaches.

4.4.2 Product-based reuse

Software reuse (namely utilization of “pre-existing software”) is addressed by requirements 7.3.4.7 (especially) and 6.5.4.16 of EN-50128. These requirements relieve the applicant from producing the same evidence as for any newly developed software, provided a number of complementary information is available (e.g., requirements to fulfil, environmental assumptions, interfaces,) and that, as a whole, the software is then accordingly and fully validated. Some requirements for the utilization of pre-existing software are graded according to SIL (e.g. identification of possible software failures, their impact on system, and implementation and validation of related counter-measures are required for SIL3, SIL4 only).

In ClearSy, for instance, reuse of system-level proofs is in focus [100]. According to their rather successful and documented experience, by defining the required properties of each subsystem composing the system and by defining the context assumptions a formal proof of system level properties is possible, even without knowing each possible vendor’s design. Such system level proof is then reused.
The ultimate goal is reuse of the rigorous reasoning that establishes that the considered system ensures its requested properties, and to assert that this reasoning is correct and fully expressed.

This reusable reasoning based on formal proofs could be exploited for the generation of fragments of assurance cases.

4.4.3 Cross-concern reuse

Given the rather recent evolution in terms of rail vehicles’ connectivity and consequent provision of standards that try to cover security aspects, no specific cross-concern and at the same time cross-domain practices seem to be existing. Nevertheless, DIN/VDE has started a proposal (last unofficial version 2015) DIN/VDE V 0831-104, “Electric signalling systems for railways – Part 104: IT Security Guideline based on IEC 62443”, on railway cybersecurity, integrating somehow EN 50128/29, EN 50159 and IEC 62443

requirements to meet the lowest Security Level (SL1) of IEC 62443 in one safety standard, thus covering basic multi-concern assurance aspects.

Recently, however, in ClearSy, concern-independent natural language proofs are investigated [100]. Thus, the reasoning proposed within these proofs could be reused when moving from one concern to another.

4.5 Space domain

ECSS-Q-ST-80C Software product assurance (6March2009), clause 7.3, lists the requirements related to software reuse. There is also a handbook to guide and given recommendations, ECSS-Q-HB-80-01A Reuse of existing software (5December2011). The standard defines existing software as:

- Any software from previous developments that is used for the project development as is or with adaptation.
- Any software including any software developed outside the contract to which ECSS software standards are applicable.
- Any software including products such as freeware and open source software products.

Reuse is based on the possibility to fully understand the existing software with respect to properties such as functionality, quality, performance, dependability and to find and adopt it to the development faster than it otherwise can be constructed.

4.5.1 Process-based reuse

In the space industry, the processes utilized for the handling of reused software are largely shared. In fact, as said before, the whole reuse process is not left to chance, rather it is thoroughly described in the aforementioned dedicated handbook (ECSS-Q-HB-80-01A, “Reuse of existing software”). Therefore, the processes that are more frequently (re-)utilized are those ensuring quality and safety of the (reused) software. Space is the only domain where the whole reuse process, as enacted in a specific project (from the initial identification of the possible pre-existing components up to their complete validation), has to be documented in a dedicated document: the Software Reuse File (SRF). The process-related documents pertaining the qualification of software development tool-chain are also re-used.

4.5.2 Product-based reuse

In the space industry, software is being adapted and re-used for the new environment, by using for instance clone-and-own practice. This usage of existing software is typically referred to as heritage instead of re-use. Typical documentation that is inherited/reused, as manually adapted, is:

- Requirements and related tracing.
- Design (and in case of code generation, code is generated from the design).
- Unit- & Integration tests are re-used and adapted manually.
- Scenario tests.
- Simulation environment.
- Parameter Engineering Database.

All such inherited documentation, as initially available and then possibly adapted, has to be explicitly cited in the aforementioned Software Reuse File (SRF).

There is no specific method to identify which components could be implemented by reusing existing software. However, differently from other standards, ECSS-Q-80C specifically asks for “...analyses of the advantages to be obtained with the selection of existing software instead of new development”. Therefore, it is asked for a sort of make or buy (reuse) trade-off analysis. The aforementioned handbook

(ECSS-Q-HB-80-01A, “Reuse of existing software”) is more precise in this respect, and clarifies the scope of this trade-off:
“The supplier should assess different options relevant to reuse and new development, evaluating them with respect to criteria such as risks, cost and benefits. These options include:

a. “Purchase of off-the-shelf, COTS software (no source code available) that satisfies the requirements
b. Develop the software product or obtain the software service internally
c. Develop the software product or obtain the external software service through contract
d. Use open source software products that satisfies the requirements
e. A combination of a, b, c and d above
f. Enhance an existing software product or service”

Again, pre-conditions, assumptions, criteria, and results of the above analysis have to be duly reported and justified, as the fundamental and leading part of the Software Reuse File (SRF). Such considerations have also to be maintained all along the project duration.

For example, within OHB Sweden, a change in a requirement that might represent the distinction between two distinct software components is handled by manually performing an impact analysis consisting of the inspection of DOORS-based requirements as well as Simulink-based design.

Lastly, it’s worth highlighting that Space too, like Automotive and Aerospace, has been recognizing already for quite some time the need to raise the level of (product) reuse and standardization in the spacecraft systems, in order to increase efficiency and reduce development costs and schedule. A number of initiatives have been started by both Industry and ESA’s R&D programmes. Those have been federated under the common “Space Avionics Open Interface Architecture” (SAVOIR) working group, organized by ESA, and with representatives from all the European Space Industry.

The overall goal of SAVOIR is to establish a streamlined on-board architecture and building blocks in order to standardize the development of on-board systems for space programmes. This reflects the need to increase efficiency and cost-effectiveness in the development process as well as account the trend towards more functionality implemented by the on-board building blocks, i.e. HW and SW components, and more complexity for the overall space mission objectives.

A specific subgroup of the on-board software reference architecture, called “SAVOIR Fair Architecture and Interface Reference Elaboration” [SAVOIR-FAIRE] working group, was spawned by SAVOIR itself. The intent was to achieve the definition of an “On-board Software Reference Architecture” [OBSW-RA], a sort of “AutoSAR” for space applications. The OBSW-RA had to be based on open interface standards, in order to allow the specification of building blocks that can be developed, qualified, and composed into compliant avionics software systems with a minimum of re-engineering effort, providing a maximum of reliability and performance, and simple to use and to implement (see [97] and [98]).

Moreover, the software architectural concept defined by CorDeT recent studies (see [98] and [99]), issued by ESA in the context of SAVIOR-FAIRE, is based on the principles of the Component Based Software Engineering (CBSE) approach. This approach defines a component model that is characterized by three software entities (3C):

- Component (which is the unit of reuse, having well-defined interfaces, explicitly captured dependencies, and which form part of its interface and purely sequential behaviour),
- Container (responsible for the realization of non-functional properties associated with tasking, synchronization and timing, using the mechanisms offered by a given execution platform)
- Connector (responsible for mediating the communication between the two components).

The approach permits the creation of software as a set of interconnected components. The set of components represents the functional architecture of the software. An underlying execution platform provides services to components, containers, and connectors. Finally, all the software is deployed on a physical architecture (computational units, equipment, and network interconnections between them).
The basic principles of the OBSW-RA identified by SAVOIR-FAIRE have been implemented and extended into multiple domains by CHESS and CONCERTO EC projects.

4.5.3 Cross-concern reuse

Within ESCC standards, cross-concern reuse is not explicitly mentioned. However, standards targeting a specific concern may warn the manufacturer to consider the other concerns.

For instance, in ECSS Q 30C [129] Section 5.7-a, it is stated "The supplier shall establish and maintain a system to track the dependability recommendations, in order to support the risk reduction process.
NOTE These recommendations are derived from the dependability analyses, and trade–off studies (typically during Phases A and B). The dependability recommendations can be tracked in combination with safety recommendations".

Cross-concern aspects are also highlighted in ECSS Q ST 40c [130] (the space product assurance-safety) Section 5.3-c "The implementation of safety requirements shall not be compromised by other requirements. NOTE For example: security requirements". Thus, the manufacturer should consider the dependencies as well as identify reuse potential by inspecting also the recently introduced standard ESSB-ST-E-008 [131].

4.6 Cross-domain

This section provides an overview of cross-domain reuse. The overview first recalls cross-domain aspects directly mentioned in the standards. Then, it offers a limited but general enough overview of state of practice.

4.6.1 Cross domain aspects within domain-specific standards

4.6.1.1 Commercial Off The Shelf (COTS)

ISO 26262 explicitly considers the cross-domain reuse. In fact, when dealing with the proven in use argument (Part 8, section 14), as generically applicable to any pre-existing item, a Note states "*For a candidate not developed in accordance with ISO 26262 (e.g. COTS products, candidates developed under a safety standard other than ISO 26262, such as IEC 61508 or RTCA DO-178), some work products of the safety case might not be available, then they are substituted by available data resulting from the development of the candidate*". Therefore, there is an implicit acknowledgment of the worth of product/process data resulting from another safety-related domain (IEC 61508 and RTCA DO-178 being mere examples).

Moreover, in the upcoming new version, as observed in [127], ISO 26262:2018-Part 11 is a new part that is expected to support semiconductor and silicon Intellectual Property suppliers on various concerns. Part 11 is also expected to be a very useful reference source for teams in the aviation industry as it expands greatly on some of the topics covered in DO-254 [110].

4.6.1.2 Tool qualification

The motivation for tool qualification as well as the overview of the standards that include tool qualification aspects are also presented in D5.1 [132]. To make this deliverable self-contained, however, we here recall and enrich the overview regarding the standards.

In automotive, ISO 26262:2011 [108] Part 8 defines three tool confidence levels (TCL 1-3) that depend on:

- Tool impact, related to the possibility that a malfunction of a particular software tool can introduce or fail to detect errors in a safety-related item or element being developed.
    - TI1 shall be selected when there is an argument that there is no such possibility.
    - TI2 shall be selected in all other cases.
- Tool error detection, related to the confidence in measures that prevent the software tool from malfunctioning and producing corresponding erroneous output, or in measures that detect that the software tool has malfunctioned and has produced corresponding erroneous output.
    - TD1 shall be selected if there is a high degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected.
    - TD2 shall be selected if there is a medium degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected.
    - TD3 shall be selected in all other cases.

Tool qualification in avionics currently is governed by the DO-330 standard [128]. It defines five tool qualification levels (TQL-1 to TQL-5), which are assigned to a given tool according to the software assurance level (A-D) and three criteria:

- Criterion 1: A tool whose output is part of airborne software and thus could insert an error.
- Criterion 2: A tool that automates verification processes and could fail to detect an error.
- Criterion 3: A tool that, within the scope of its intended use, could fail to detect an error.

As a rule of thumb, Criterion 3 is related to computer-aided specification, Criterion 2 to V&V tools, and Criterion 1 to compilers.

For railway application, the only relevant standard for tool qualification is EN 50128:2011 [103], which provides engineering assistance tool requirements. Three tool classes are defined:

- T1, which is related to specification assistance (no safety impact in case of errors in this tool).
- T2, which is related to tools that if an error occurs, a safety requirement may be missed.
- T3, which is related to safe data computation.

The classes are therefore similar to those from avionics: T1 is related to computer-aided specification, T2 is related to test automation tools, and T3 to compilers.

It should be noted that ISO 26262 (Part 8:11.4.6) states that a tool developed according to the DO330 standard can be considered sufficient for being suitable for ISO 26262 ASIL-D projects.

Comparative work at tool qualification process level for the purpose of reuse of process information was conducted in the framework of the SafeCer project by Gallina et al. [27].

For the purpose of AMASS cross-domain case study, the comparison between automotive and avionics is sufficient. A broader comparative study was offered in [138], where a larger number of standards where discussed.

4.6.2 Cross domain state of practice

Within the state of practice's world, comparative studies [133-139] to highlight similarities and differences, have been conducted in France by the Working Group "Safety Standards", created in 2008, and initially attached to the "Club des Grandes Entreprises de l'Embarqué (CG2E)".

Outside Europe, Cunha Trivelato et al [113] highlights that the aircraft, rail, and automobile industry handles the same main function, namely: "To transport a (set of) person or a cargo safe and with comfort from point A until point B". Given this commonality, it would "make sense for a country, a group of countries or even all countries to unify the certification agencies as well as the applicable standards" or at least to cooperate for the identification and reuse-based exploitation of their mappings.

In the remaining part of this section, the focus is on reuse between two specific domains: avionics and automotive. The reason for choosing these two domains is because of the current trends [113].

In the past, we witnessed the birth of the x-by -wire system in the automotive domain, bringing together the concepts from Aerospace and Aeronautic industries. Today, we are witnessing an inverted process of diffusing the technology, i.e., the technology developed by the automotive industry is migrating to the aeronautic industry, e.g., the COTS, providing reduction of costs and shorter periods of development, forming a robust solution of engineering.

In the past, as discussed by da Cunha Trivelato et al [113], the differences between the automotive and the avionics domain were more powerful than their commonalities, preventing them from cooperating practically on any level of products (from components to standards), in any phase of the life cycle processes (from capture of needs to disposal/retirement), in any kind of organization (from requirements teams to test facilities), or in any certification authorities and procedures. Today, their commonalities are becoming more powerful than their differences. Thus, the automotive and the avionics industries are interested in cooperating on many levels of products, in some phases of the life cycle processes, in some parts of organizations, and even in some certification authorities and procedures.

Given the rather recent interest in cooperation, well-established methods for cross-domain reuse are not available on the shelf. However, some practices are in use.

4.6.3 Process-based reuse

Da Cunha Trivelato et al [113], in the context of highly integrated IMA systems, envision an integral process that supports system design consistence, safety and security requirements allocated to each system, robust portioning avoiding adverse effects, health monitoring and fault management, configuration management, human factors requirements, and certification requirements. The right way to develop an IMA-compliant component, application or system is to focus the organization on process and process improvement instead of mere components or parts. The integral process is derived from considering the interplay between major techniques and processes mandated by the avionics and automotive standards.

4.6.4 Product-based reuse

In the components area, again according to da Cunha Trivelato et al [113], the most important convergence between the avionics and automotive domains will be the reusable computational platform based on a standard like Integrated Modular Avionics (IMA). The aircraft industry will benefit from the scalability in the automotive industry while automotive industry will benefit from computational power concentration and the safe critical certification process from avionics.

Within the AMASS project, Infineon is particularly interested in cross-domain reuse. Infineon for instance states, that the quality of the vehicles they buy, as consumers, and the cost competitiveness of the automotive industry depend on the product developers being able to make quality and reliability predictions. To bring products into volume production, qualification measures are necessary. These qualification measures shall provide useful and accurate data to verify that developers are producing meaningful results for the reliability of their products under defined Mission Profiles from the whole supply chain.

The qualification strategy known since years in the avionic industry is now entering the automotive area. This means that individual stress test driven qualification standards (see [94] and [95]) are no longer that relevant because finally it is the projection of the reliability data on the mission profile, which is necessary for an assessment of sufficient reliability. By using reliability models, a transfer of qualification results is possible for different mission profiles like automotive or aviation. It is no longer the predefined stress test

that is different; it is the extrapolation parameter, which is used in the reliability model that is different. Only some of these tests can be performed on product level. For less critical applications, it is sufficient to perform individual stress test driven qualification. A decision process to choose the appropriate qualification strategy is based on requirements, intended application and already existing data, which could be reused. The decision on the appropriate method has to be made on the level of the individual failure mechanism. The focus has shifted to reliability methodologies applied on technology or process level during development.

Now product qualification changes: from the detection of defects during stress tests (based on predefined sample sizes) to the generation of knowledge by generating failure mechanisms specific data, combined with the knowledge from the technology field. Now we can generate knowledge about the robustness of products. Robustness Validation (RV) [96] is a "top down" process to demonstrate the robustness of a semiconductor component under a defined Mission Profile. RV represents an approach to qualification and validation that is based on knowledge of failure mechanisms and relates to specific Mission Profiles. The knowledge gained by applying this approach leads to improvement that extends beyond the component and its manufacturing process under consideration. RV contains great potential for re-use, which contributes in its entirety to a significant increase in quality and reliability, time to market and reduction of costs. Last, but not least, this will result in improvement of the competitiveness of all involved participants in the value adding chain.

A Mission Profile defines the conditions of use for the component in the intended application. The Mission Profile establishes the basis for the RV approach, providing necessary additional information that is not described in the datasheet. The knowledge generated during a qualification for an automotive specific mission profile could be used to assess the reliability under an avionic related mission profile. These qualification data could also be used to forecast whether a stress test required by an avionic standard would be passed. This means that knowledge generated within an automotive qualification could be re-used to assess the coverage required by an automotive standard.

4.6.5 Cross-concern reuse

Given the rather recent evolution in terms of vehicles' connectivity and consequent provision of standards that try to cover security aspects, no specific cross-concern and at the same time cross-domain practices seem to exist yet. With the existing and upcoming standardization schemes in both domains, it is evident that cross-concern reuse will be an issue to be solved.

4.6.6 Reuse Approach and Possibilities with RQS

As an example of the reuse support offered by the current tools available in the market, this section presents the kind of reuse that can be performed with RQS, a tool suite developed by the The Reuse Company for ontology-based Requirements Quality.

The RQS tools work based on a master database, which ease information exchange between the stakeholders in the processes of requirements authoring, quality analysis and knowledge management. This approach of sharing information, supports the different stakeholders (players) involved in the process of quality improvement and knowledge management in their work oriented to achieve a common goal.

One of the first topics to address when deciding how to build a master database for RQS is the working environment. This issue is specifically related to defining the database deployment, between one single database for the whole company, or several databases following one or many different split dimensions.

Table 1 summarizes a series of pros and cons for both approaches ("+" sign for positive parameter, "–" for negative parameter, and a number in brackets to identify the list of reasons below):

Table 1: One ontology vs. several ontologies

Parameter	One database	Several databases
Common Vocabulary	+(1)	-
Common Cluster definition	+ (1)	-
Specific vocabulary	- (2)	+ (2)
Common tokenization & disambiguation rules	+ (3)	- (3)
Common patterns	+ (4)	- (4)
Common quality rules	+ (5)	- (5)
Ease of use	= (6)	= (6)
Ease of evolution	- (7)	+ (7)
Access management	- (8)	+ (8)

(1) Even if it is not so clear that every specific term has a consistent meaning all across the organization.

(2) Those domain specific terms, that are only used in a specific domain, or whose definition, clustering, links… are different from one domain to another, are easier to be managed in several different databases.

(3) Sometimes it is not easy to find different rules in different business domains. In this case, a centralized repository eases the process of maintaining this information.

(4) As in the previous case, the definition of the requirements patterns is pretty much the same across different technical domains.

(5) Quality rules should be the same across the whole organization. Of course, RQS shall be adapted to different levels of maturity in the teams, different levels of risk in the projects… Thus, the baseline configuration should be the same across the organization, while each requirements document, for each different project, must be assigned to a particular quality baseline.

(6) Common content is managed more easily with one single database, while specific content is managed more easily with several databases. However, RQS provides means to share information about different databases (SKR – System Knowledge Repository). See Figure 19 as an example on how to evolve a common SKR starting with several small contributors.

(7) Clearly the easiest way to manage the evolution of specific items of the overall SKR is by means of different small databases, where these changes can be agreed and tested in a closer environment. After that, when the changes are agreed, they can be pushed into the global repository.

(8) Access management, especially in terms of rights to edit information, is more easily managed in a context with different databases.

Given what is stated above, the recommendation from The REUSE Company is to use a centralized repository, containing all the common items of the Knowledge Repository. Moreover, the specific information evolves in different small repositories (contributors) and is then shared with the main (master) SKR using the mechanisms described below.

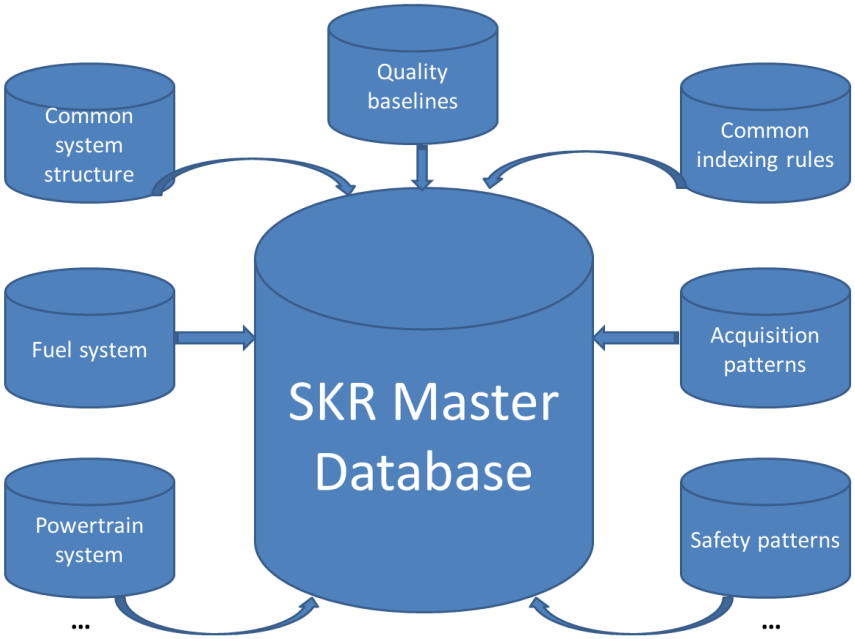

Figure 19: Database ecosystem (based on contributors)

All these processes requested to evolve each RQS Database contributor following its specific evolution line over the time, and eventually merge its content into the RQS master database, are now enforced thanks to some new capabilities of Knowledge Manager that can be summarized in the following highlights:

- Ontology baselining
- Baselines compare algorithm
- Ontology compare algorithm
- Reverse to baseline
- Advanced ontology merging
- Advanced ontology copy

The reader may refer to the RQS help file and other RQS manuals on how to use all the previous topics.

As for most of the work-products in a project, the different versions (evolution) of the System Knowledge Base should be baselined. Once a particular baseline is agreed and tested, it can become part of the production environment.

For this reason, as depicted in Figure 20, TRC proposes the definition of three main zones together with a synchronization mechanism between zones. These three zones are:

Figure 20: Definition of zones for production ontology

Zone 3: Development

Zone 3 is the zone conceived to work with all the different contributors (building blocks) for the future production database: different groups of patterns, different sub-systems definition, etc. Each of these contributor databases may have a specific ontology owner. All contributors, as well as the main master database, are present in this zone, and accessible by means of the Knowledge Manager (KM). There is no need of a Requirements Authoring Tool (RAT) in this zone. The common aspects can be managed in the master SKR database, while the specific topics are managed by the small contributors.

It should be noted that one very important input in the definition of several aspects of the SKR must come from the production environment. Synchronization mechanisms are used to define and enable communication mechanisms among zones.

Zone 2: Test

Zone 2 represents a transient stage for the ontologies where consistency shall be guaranteed before its transition to the production zone. In this zone, all the contributors are merged to generate the unique SKR database that will be eventually pushed into the production zone. The RQS Clients (RQA and RAT) should be used in this zone to validate the content of the resulting master SKR.

Zone 1: Production

Zone 1 consists of the real production database. The RQS Server shall be connected to this database so that all the RQS clients (i.e. RQA and RAT) can access and use this master database.

Following the database ecosystem described in Figure 19, this zone should only contain the master database, and there is no need to duplicate all small contributors into the production zone.

The following synchronization aspects need to be taken into account:

Common synchronization process

Each of the contributors to the master database evolves separately in Zone #3. In order to do so, the brand new capability of KM to deal with baselines will be highly appreciated.

Once changes are ready from the different contributors for production, it's time to push information from zone #3 to zone #1. However, this may imply:

- To take into consideration also information already in zone #1, but not of the different contributors (zone #3).
- To test that all the new information coming from different contributors is not conflicting when joined together.

Therefore, an intermediate zone #2 is designed just to join all this information together, and test that everything is fine before swapping with the production database (zone #1).

The possible conflicting information may include:

- Terms in different forms (e.g. singular and plural) coming from different contributors.
- Conflicting definitions of terms (i.e. term notes, clusters…).
- Conflicting relationships between terms.
- Conflicting tokenization/disambiguation rules that may incur an ill-defined indexing process.
- Conflicting patterns (because of the pattern weight).

Therefore, the zone #2 working process is the following:

1. The different contributors are merged together.
2. The merge logs are analysed by the owner of the zone[4].
3. An ontology consistency sub-process shall be performed to guarantee a coherent ontology ready to swap into zone #1.
4. Import from zone #1: document mapping with quality baselines and specific quality metrics for specific documents.
5. Swapping with zone #1.

Managing suggestions across different zones

Unfortunately, there are some processes that imply crossing boundaries between the different zones. A clear example is the evolution of the ontology contributors based on the suggestions provided by the RQS users using zone #1.

Therefore, the suggestion mechanism should be pushed to a new level in this production approach. Now the process is the following:

1. RQA and RAT users issue change requests.
2. The ontology (zone #1) owner checks the requests, and can revoke some suggestions (the process goes no further for such revoked suggestions).
3. The preliminary accepted suggestions shall now be considered by the real “owner” of this information, that is, the owner of the contributor ontology. Thus, KM shall be able to issue change requests from the master database to a specific contributor.
4. The owner of the contributor database may revoke or accept. In both cases, the owner of the master database is notified.
5. In case of an accepted request, the owner of the contributor database may:
    a. Proceed with the changes in the contributor database (zone #3).
    b. Push these changes to the master database as well.

[4] Probably the same role as the owner of the master database

5. Consolidation and way forward

This chapter aims at indicating a way forward to enable a co-evolution of the state of the art together with the state of practice. The way forward is based on the findings obtained via the investigation of the current status in both academia and industry. This chapter is structured in three main sections that reflect the three main types of expected reuse: process, product, and assurance case.

5.1 Consolidation and way forward concerning process

A considerable set of normative documents contains guidance on processes: development processes, tool qualification processes, documentation processes, etc. From the survey of such normative documents, it clearly emerges that the provision of process-related information is mandatory or highly recommended in all the domains of interest in the context of AMASS. It also appears trivial that reuse of such information would reduce cost and would also save time to be invested in more significant activities related to e.g., verification of a product's behaviour.

From the survey of the literature and current practices, it emerges that there are neither well-established process languages, nor mature corresponding tool support for enabling systematic reuse of process elements/structures in the context of safety-critical systems in the various domains.

In the context of the SafeCer project, SPEM2.0 and its reference-implementation represented by the Eclipse-based EPF Composer were used. EPF Composer was customized to demonstrate as a proof of concepts the benefits of Safety-Oriented Process Line Engineering in terms of reuse of process elements. Based on the assumption that the criticality levels differ in terms of process-stringency, the lowest process elements were expected to be incremented (via the *contributes* relationship) and made vary to define process elements of higher criticality. The criticality level was thus represented implicitly. In the context of the OPENCOSS project, a new meta-model for modelling processes was introduced and a specific tool was developed. Via the OPENCOSS-tool, different but similar processes were put in relation via maps.

A deeper analysis of the correspondence between the approaches developed in SafeCer, OPENCOSS, and other relevant projects together with a deeper analysis of the needs stemming from the AMASS case studies, will allow the AMASS project to identify the concepts necessary for process-based compliance and for process-related reuse in a wide range of scenarios, larger than those addressed separately by SafeCer and OPENCOSS. These concepts will be determined in the scope of T6.2 and will be part of CACM. Based on the initial analysis conducted for this deliverable, we expect that some concepts will already be supported by both SafeCer and OPENCOSS, and others by only one of the approaches. It is also possible that concepts that are not supported by any of the approaches will become part of CACM, as a result of the need for providing reuse support for further application domains and for further industrial scenarios in AMASS. Within T6.2 in cooperation with T3.2, it will be also investigated if the modelling solution conceived for process-oriented normative documents could also be used for other kinds of normative documents.

The proper modelling and management of variability (including process structures) remains an open issue in both approaches. From a strategic point of view, however, despite the current silence of the community around EPF Composer, this latter seems to represent the right tool to work on. The main reason is its support for reusable process elements and its interesting and profitable connection with the Rational Method Composer [92], which is already used for compliance management [93]. More specifically, the work that we envision within Task 6.2 and Task 6.3 represents an evolution of EPF Composer, aimed at supporting a more intuitive variability management.

5.2 Consolidation and way forward concerning product

From the survey of normative documents, it clearly emerges that the provision of product-related information is also mandatory or highly recommended in all the domains of interest in the context of this project. It also emerges that the amount of the mandatory or highly recommended information to be provided is huge and cannot be provided from scratch for each product. Normative documents embrace requirements for enabling reuse of product-related information.

The overview of the state of practice related to product's manufacturers has revealed the presence of in-house solutions aimed at reusing information. Very often these solutions are not systematic. Moreover, they lack formality and support for automation. Some tool providers, instead, seem to be aligned with the recent development in the Semantic Web and offer ontology-based solutions for enabling reuse.

The overview of the state of art has revealed the presence of promising approaches to systematize the reuse of product-based information. These approaches are: contract-based reasoning, patterns, model-based principles applied to engineering and product certification, and variability management. However, these approaches need, in some cases, to be further developed and integrated.

The work that we envision within Task 6.2 is an integration and further extensions of such approaches.

5.3 Consolidation and way forward concerning assurance cases

From the survey of normative documents, it clearly emerges that the provision of assurance cases is also mandatory or highly recommended in all the domains of interest in the context of this project.
Assurance cases have to be well-founded and explained. To be well-founded, appropriate evidence must be provided. To be explained, the reasoning/argumentation, which is expected to connect the claims with the evidence, must be developed. Reuse in the context of assurance cases consists of reuse of reasoning/argumentation-fragments together with the appropriate evidence. Reuse of evidence (immediate, direct, and indirect) was already discussed in the previous sections of this chapter.

Reuse of reasoning/argumentation is rather deeply and broadly explored in the state of the art via patterns, variability management, contract-based, and module-based argumentation approaches as well as model-based argumentation approaches. However, in the state of practice, such solutions for enabling reuse of reasoning are not yet adopted due to the rather recent introduction of requirements related to the provision of assurance cases (see ISO 26262-related requirements on safety cases) or rather recent increased awareness of the presence of implicit requirements related to assurance cases in avionics.
As discussed, the railway domain represents an exception. Within railway industry, given the rather well defined and structured definition of a safety case within EN 50129, reusable reasoning is explored and proposed.

The work that we envision within Task 6.2 is thus to empower the current trend within railways and to spread it to the other domains addressed in AMASS. The natural language-based reusable proof could and should benefit from current developments in the state of the art in terms of argumentation languages, patterns, and variability modelling. The envisaged evolution of the CCL in accordance with SACM is expected to enable a semi-formal representation of the reusable proof. Moreover, via model-based argumentation approaches, horizontal model transformations will enable the generation of different representations depending on the user.

Abbreviations and Definitions

ATL	ATLAS Transformation Language
BPMN	Business Process Model and Notation
BVR	Base Variability Resolution
CACM	Common Assurance and Certification Meta-model
CBSE	Component-Based Software Engineering
CCL	Common Certification Language
COTS	Commercial Off The Shelf
CPS	Cyber Physical Systems
CVL	Common Variability Language
DSL	Domain Specific Language
GSN	Goal Structuring Notation
IMA	Integrated Modular Avionics
MDE	Model Driven Engineering
MOF	Meta Object Facility
MOTS	modified off the shelf
OMG	Object Management Group
OTS	off the shelf
PLE	Process Line Engineering
RQS	Requirements Quality Suite
SACM	Structured Assurance Case Meta-model
SEI	Software Engineering Institute
SEooC	Safety Element out-of-Context
SKR	System Knowledge Repository
SOUP	Software of Unknown Pedigree
SPEM	Software and Systems Process Engineering Meta-model
UDP	User-defined Process

References

[1] Kelly, Tim P., and John A. McDermid. "Safety case construction and reuse using patterns." *SafeComp 97*. Springer London, 1997. 55-69.

[2] Ye, Fan. *Justifying the use of COTS Components within safety critical applications*. University of York, 2005.

[3] Alexander, Robert, et al. *Safety cases for advanced control software: Safety case patterns*. University of York, Department of Computer Science, 2007.

[4] Wu, Weihang. *Architectural reasoning for safety-critical software applications*. University of York, Department of Computer Science, 2007.

[5] Palin, Robert, and Ibrahim Habli. *Assurance of automotive safety–A safety case approach*. SafeComp. Springer, 2010.

[6] Ayoub, Anaheed, et al. "A safety case pattern for model-based development approach." *NASA Formal Methods*. Springer Berlin Heidelberg, 2012. 141-146.

[7] Conmy, Philippa, and Iain Bate. "Assuring safety for component based software engineering." *High-Assurance Systems Engineering (HASE), 2014 IEEE 15th International Symposium on*. IEEE, 2014.

[8] Armengaud, Eric. "Automated safety case compilation for product-based argumentation." *Embedded Real Time Software and Systems (ERTS)(February 2014)* (2014).

[9] Basir, Nurlida, Ewen Denney, and Bernd Fischer. "Building heterogeneous safety cases for automatically generated code." *Infotech@ Aerospace Conference. AIAA*. 2011.

[10] Denney, Ewen, and Ganesh Pai. "Automating the assembly of aviation safety cases." *Reliability, IEEE Transactions on* 63.4 (2014): 830-849.

[11] Prokhorova, Yuliya, Linas Laibinis, and Elena Troubitsyna. "Facilitating construction of safety cases from formal models in Event-B." *Information and Software Technology* 60 (2015): 51-76.

[12] Habli, Ibrahim, and Tim Kelly. "A safety case approach to assuring configurable architectures of safety-critical product lines." *Architecting Critical Systems*. Springer Berlin Heidelberg, 2010. 142-160.

[13] Gallina, Barbara, et al. "VROOM & cC: a method to build safety cases for ISO 26262-compliant product lines." *SAFECOMP 2013-Workshop SASSUR (Next Generation of System Assurance Approaches for Safety-Critical Systems) of the 32nd International Conference on Computer Safety, Reliability and Security*. 2013.

[14] Schulze, Michael, Jan Mauersberger, and Danilo Beuche. "Functional safety and variability: can it be brought together?." *Proceedings of the 17th International Software Product Line Conference*. ACM, 2013.

[15] Hutchesson, Stuart, and John McDermid. "Trusted product lines." *Information and Software Technology* 55.3 (2013): 525-540.

[16] Latour, Larry, Tom Wheeler, and Bill Frakes. "Descriptive and predictive aspects of the 3Cs model: SETA1 working group summary." *ACM SIGAda Ada Letters*. Vol. 11. No. 3. ACM, 1991.

[17] Kath, Olaf, Rudolf Schreiner, and John Favaro. "Safety, security, and software reuse: a model-based approach." *Proceedings of the Fourth International Workshop in Software Reuse and Safety*. 2009.

[18] Fenn, Jane L., et al. "The who, where, how, why and when of modular and incremental certification." *System Safety, 2007 2nd Institution of Engineering and Technology International Conference on*. IET, 2007.

[19] Ye, Fan. *Justifying the use of COTS Components within safety critical applications*. University of York, 2005.

[20] Šljivo, Irfan. "Facilitating Reuse of Safety Case Artefacts Using Safety Contracts." Licentiate Thesis. Mälardalen University. (2015).

[21] Chen, DeJiu, et al. "Modelling support for design of safety-critical automotive embedded systems." *Computer Safety, Reliability, and Security*. Springer Berlin Heidelberg, 2008. 72-85.

[22] Habli, Ibrahim. *Model-based assurance of safety-critical product lines*. University of York, 2009.

[23] Hawkins, Richard, et al. "Weaving an assurance case from design: a model-based approach." *High Assurance Systems Engineering (HASE), 2015 IEEE 16th International Symposium on*. IEEE, 2015.

[24] Hawkins, R., I. Habli, and T. Kelly. "The Need for a Weaving Model in Assurance Case Automation." Ada User Journal, Volume 36, No. 3 pp. 187 - 191, September 2015

[25] Gallina, B.: A Model-driven Safety Certification Method for Process Compliance. In: 2nd International Workshop on Assurance Cases for Software-intensive Systems. pp. 204–209. IEEE (Nov 2014)

[26] B. Gallina, E. Gomez-Martınez, and C. Benac Earle. Deriving Safety Case Fragments for Assessing MBASafe's Compliance with EN 50128. 16th International SPICE Conference on Process Improvement and Capability dEtermination (SPICE), Dublin, Ireland, Vol. 609, Communications in Computer and Information Science series, Springer, 2016.

[27] B. Gallina, S. Kashiyarandi, K. Zugsbrati and A. Geven. *Enabling Cross-domain Reuse of Tool Qualification Certification Artefacts*. Proceedings of the 1st International Workshop on DEvelopment, Verification and VAlidation of cRiTical Systems (DEVVARTS), joint workshop at SafeComp conference, Springer, LNCS 8696, ISBN: 978-3-319-10556-7, pp. 255-266, Florence (Italy), 8 September, 2014.

[28] B. Gallina, I. Sljivo, and O. Jaradat. Towards a Safety-oriented Process Line for Enabling Reuse in Safety Critical Systems Development and Certification. Post-proceedings of the 35th IEEE Software Engineering Workshop (SEW-35), IEEE Computer Society, ISBN 978-1-4673-5574-2, Heraclion, Crete (Greece), 2012.

[29] B. Gallina, S. Kashiyarandi, H. Martin and R. Bramberger. *Modeling a Safety- and Automotive-oriented Process Line to Enable Reuse and Flexible Process Derivation*. Proceedings of the 8th IEEE International Workshop on Quality-Oriented Reuse of Software (QUORS), joint workshop at COMPSAC conference, IEEE Computer Society, doi: 10.1109/COMPSACW.2014.84, pp. 504-509,Västerås (Sweden), 2014.

[30] B. Gallina, K. Lundqvist and K. Forsberg. *THRUST: A Method for Speeding Up the Creation of Process-related Deliverables*. IEEE 33rd Digital Avionics Systems Conference (DASC-33), doi:10.1109/DASC.2014.6979489, Colorado Springs, CO, USA, October 5-9, 2014.

[31] B. Gallina. *Towards Enabling Reuse in the Context of Safety-critical Product Lines*. 5th International Workshop on Product LinE Approaches in Software Engineering (PLEASE), joint event of ICSE, Florence, Italy, May 19th, 2015.

[32] B. Gallina, Z. Szatmari. *Ontology-based Identification of Commonalities and Variabilities among Safety* Processes. Proceedings of the 16th International Conference on Product-Focused Software Process Improvement (PROFES), Springer, LNCS, Bolzano, Italy, December 2-4, 2015.

[33] B. Gallina, L. Fabre. Benefits of Security-informed Safety-oriented Process Line Engineering. IEEE 34th Digital Avionics Systems Conference (DASC-34), Prague, Czech Republic, September 13-17, 2015.

[34] "Certifying boeing's airplanes," http://787updates.newairplane.com/Certification-Process, accessed: 2016-07-27.

[35] O. Jaufman and J. Münch, Acquisition of a Project-Specific Process. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 328–342.

[36] M. Kuhrmann, D. M. Fernández, and T. Ternité, "Realizing software process lines: Insights and experiences," in Proceedings of the 2014 International Conference on Software and System Process, ser. ICSSP 2014. New York, NY, USA: ACM, 2014, pp. 99–108.

[37] O. M. Group, "Software & systems process engineering meta-model specification," Object Management Group, Tech. Rep. formal/2008-04-01, April 2008. [Online]. Available: http://www.omg.org/spec/SPEM/2.0/PDF

[38] J. A. Hurtado Alegría, M. C. Bastarrica, A. Quispe, and S. F. Ochoa, "An mde approach to software process tailoring," in Proceedings of the 2011 International Conference on Software and Systems Process, ser. ICSSP '11. New York, NY, USA: ACM, 2011, pp. 43–52.

[39] T. Martínez-Ruiz, F. García, and M. Piattini, Towards a SPEM v2.0 Extension to Define Process Lines Variability Mechanisms. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 115–130.

[40] J. Simmonds and M. Bastarrica, "Modeling variability in software process lines," Universidad de Chile, Tech. Rep. TR/DCC-2011-10, 2011.

[41] K. C. Kang, J. Lee, and P. Donohoe, "Feature-oriented product line engineering," IEEE Software, vol. 19, no. 4, pp. 58–65, Jul 2002

[42] E. A. Oliveira Junior, M. G. Pazin, I. M. S. Gimenes, U. Kulesza, and F. A. Aleixo, SMartySPEM: A SPEM-Based Approach for Variability Management in Software Process Lines. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 169–183.
[43] F. A. Aleixo, M. A. Freire, W. C. dos Santos, and U. Kulesza, Automating the Variability Management, Customization and Deployment of Software Processes: A Model-Driven Approach. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 372–387.
[44] "Eclipse process framework project (epf)," https://eclipse.org/epf/, accessed: 2016-07-12.
[45] "Genarch model based derivation tool," http://www.dcomp.ufsj.edu.br/ elder/genarch/index.html, accessed 2016-07-12.
[46] "jbpm - open source business process management - process engine," http://www.jbpm.org/, accessed: 2016-07-12.
[47] E. Rouillé, B. Combemale, O. Barais, D. Touzet, and J. M. Jézéquel, "Leveraging cvl to manage variability in software process lines," in 19th Asia-Pacific Software Engineering Conference, vol. 1, Dec 2012, pp. 148–157.
[48] J. Simmonds and M. Bastarrica, "Modeling variability in software process lines," Universidad de Chile, Tech. Rep. TR/DCC-2011-10, 2011.
[49] J. A. H. Alegría and M. C. Bastarrica, "Building software process lines with casper," in International Conference on Software and System Process (ICSSP), June 2012, pp. 170–179.
[50] F. Jouault and I. Kurtev, Transforming Models with ATL. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 128–138.
[51] J. Friedrich, M. Kuhrmann, M. Sihling, and U. Hammerschall, Das V-Modell XT. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–32.
[52] M. Kuhrmann, D. M. Fernández, and T. Ternité, "Realizing software process lines: Insights and experiences," in Proceedings of the 2014 International Conference on Software and System Process, ser. ICSSP 2014. New York, NY, USA: ACM, 2014, pp. 99–108.
[53] J. Schramm, P. Dohrmann, and M. Kuhrmann, "Development of flexible software process lines with variability operations: A longitudinal case study," in Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, ser. EASE '15. New York, NY, USA: ACM, 2015, pp. 13:1–13:10.
[54] C. Ayora, V. Torres, B. Weber, M. Reichert, and V. Pelechano, "Vivace: A framework for the systematic evaluation of variability support in process-aware information systems," Information and Software Technology, vol. 57, pp. 248 – 276, 2015.
[55] M. L. Rosa, W. van der Aalst, M. Dumas, and F. Milani, "Business process variability modeling: A survey," Queensland University of Technology, Tech. Rep., 2013.
[56] C. Ayora, V. Torres, J. L. de la Vara, and V. Pelechano, "Variability management in process families through change patterns," Information and Software Technology, vol. 74, pp. 86 – 104, 2016.
[57] A. Hallerbach, T. Bauer, and M. Reichert, "Capturing variability in business process models: The provop approach," J. Softw. Maint. Evol., vol. 22, no. 67, pp. 519–546, Oct. 2010.
[58] A. Schnieders and F. Puhlmann, Variability Modeling and Product Derivation in E-Business Process Families. Dordrecht: Springer Netherlands, 2007, pp. 63–74.
[59] J. L. de la Vara and R. K. Panesar-Walawege, SafetyMet: A Meta-model for Safety Standards. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 69–86.
[60] A. Hallerbach, T. Bauer, and M. Reichert, "Capturing variability in business process models: The provop approach," J. Softw. Maint. Evol., vol. 22, no. 67, pp. 519–546, Oct. 2010.
[61] "Aris architect," http://www.ariscommunity.com/university/downloads/aris-business-architect, accessed: 2016-07-14
[62] "The pesoa project, process families as core area of research," https://delta-software.com/en/we/research/projects/pesoa-project.html, accessed: 2016-07-14
[63] "Hypersenses - simply tailor made," https://delta-software.com/en/our-offer/products/hypersenses.html, accessed: 2016-07-14.
[64] K. Czarnecki and M. Antkiewicz, Mapping Features to Models: A Template Approach Based on Superimposed Variants. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 422–437

[65] "fmp2rsm: Mapping features to uml 2.0 models plug-in," http://gsd.uwaterloo.ca/fmp2rsm, accessed: 2016-07-15.
[66] O. Haugen, "Common variability language (cvl)," Object Management Group, Tech. Rep. ad/2012-08-05, August 2012. [Online]. Available: http://www.omgwiki.org/variability/doku.php
[67] "Bvr tool," http://modelbased.net/tools/bvr-tool/, accessed: 2016-07-27.
[68] "Varies," http://www.varies.eu/, accessed: 2016-07-27.
[69] S. Nair, N. Walkinshaw, T. Kelly, and J. L. de la Vara, "An evidential reasoning approach for assessing confidence in safety evidence," in *IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*, Nov 2015, pp. 541–552.
[70] Modsafe, http://www.modsafe.eu/, accessed: 2016-07-27.
[71] Lautiere, S., D., Cooper, and D., Jackson, February 2005, SafSec: Commonalities Between Safety and Security Assurance, Proceedings of the 13th Critical Systems Symposium, Southampton, England.
[72] The MAFTIA project, http://research.cs.ncl.ac.uk/cabernet/www.laas.research.ec.org/maftia/
[73] Praxis High Integrity Systems, SafSec: Integration of Safety & Security Certification, November 2006.
[74] Lautiere, S., D., Cooper, D., Jackson, T., Cockram, 2004, Assurance Cases: how assured are you?, supplemental volume to DSN-2004, Proceedings of the International Conference on Dependable Systems and Networks..
[75] Dobbing, B., S., Lautiere, 2007, Dependability-by-Contract. In The Safety of Systems (pp. 35-51). Springer London.
[76] J. Rushby. Just-in-time certification. In proc. of the 12th IEEE International Conference on Engineering Complex Computer Systems,pp. 15–24, 2007.
[77] I. Dodd and I. Habli, "Safety certification of airborne software: An empirical study", Reliability Engineering & System Safety, vol. 98, no. 1, pp. 7–23, Feb. 2012.
[78] Introduction to IS0, 1997. http://www. iso. ch/infoe/intro.html
[79] W. Emmerich, A. Finkelstein, C. Montangero, S. Antonelli, S. Armitage and R. Stevens. Managing Standards Compliance, IEEE Transactions on Software Engineering. Vol. 25. No.6. pp.836-851. Nov-Dec 1999.
[80] EVOLVE project: http://www.evolve-itea.org (Accessed Feb 8, 2012)
[81] ModelME! project: http://modelme.simula.no/ (Accessed Feb 7, 2012)
[82] Paul W.H. Chung, Larry Y.C. Cheung and Colin H.C. Machin. Compliance Flow - Managing the Compliance of Dynamic and Complex Processes. Journal of Knowledge Based Systems. Vol. 21. No.4. pp. 332-354. May 2008.
[83] J. L. de la Vara; A. Ruiz; K. Attwood; H. Espinoza; R. Kaur Panesar-Walawege; Á. López; I. del Río; T. Kelly, "Model-based specification of safety compliance needs for critical systems: A holistic generic meta-model", Information & Software Technology 72: 16-30, 2016.
[84] E. Stensrud, T. Skramstad, J. Li and J. Xie. Towards Goal-based Software Safety Certification Based on Prescriptive Standards, International Workshop on Software Certification (WoSoCER), 2011.
[85] A. Ruiz, "A Harmonized Compositional Assurance Approach for Safety-Critical Systems", Ph.D. Thesis, 2015.
[86] Zeller M, Höfig K, Rothfelder M. Towards a Cross-Domain Software Safety Assurance Process for Embedded Systems. SASSUR 2014, pp. 396–400.
[87] Papadopoulos Y, McDermid JA. The potential for a generic approach to certification of safety critical systems in the transportation sector. Reliab. Eng. Syst. Saf. 63(1):47-66, 1999.
[88] E. Denney y G. Pai, "A Lightweight Methodology for Safety Case Assembly", in Computer Safety, Reliability, and Security, vol. 7612, F. Ortmeier y P. Daniel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1-12.
[89] A. Ruiz, H. Espinoza, F. Tagliablò, S. Torchiaro, A. Melzi, "A Preliminary Study towards a Quantitative Approach for Compositional Safety Assurance" Proceedings of 21st Safety Critical Systems Symposium, February 2013.
[90] A. Ruiz, I. Habli, and H. Espinoza, 'Towards a Case-Based Reasoning Approach for Safety Assurance Reuse', in Computer Safety, Reliability, and Security, F. Ortmeier and P. Daniel, Eds. Springer Berlin Heidelberg, 2012, pp. 22–35.

[91] M. Bender, T. Maibaum, M. Lawford, A. Wassyng, Positioning Verification in the Context of Software/System Certification. In Proceedings of the 11th International Workshop on Automated Verification of Critical Systems (AVoCS 2011), Electronic Communications of the EASST (European Association of Software Science and Technology), Volume 46, 2013.

[92] IBM® Rational® Method Composer https://www.ibm.com/support/knowledgecenter/SSBSK5_7.5.2/com.ibm.rmc.help.doc/topics/a_product_overview.html

[93] G. Bleakley. How Rational can help with compliance to ISO 26262 and ASPICE, IBM Corporation, 2014. https://www-01.ibm.com/events/wwe/grp/grp008.nsf/vLookupPDFs/Graham%20Bleakley%20-%20ISO%2026262%20-ASPICE%20General%20Overview/$file/Graham%20Bleakley%20-%20ISO%2026262%20-ASPICE%20General%20Overview.pdf

[94] AEC Q100, Failure Mechanism Based Stress Test Qualification for Integrated Circuits.

[95] AEC Q101, Stress Test Qualification for Discrete Semiconductors.

[96] Handbook for Robustness Validation of Semiconductor Devices in Automotive Applications (3rd edition), ZVEI 2015.

[97] SAVOIR-FAIRE Working Group, “Space on-board software reference architecture”, Proceedings of DASIA Conference, Budapest, May 2010.

[98] SAVOIR-FAIRE Working Group, SAVOIR-FAIRE – On-board software reference architecture, Issue 1, TECSWE/09-289/AJ, 10 June 2010.

[99] CORDET 3 Study, On-board Software Reference Architecture: Specification - D02-OSRA-SPEC, Issue 1.0, 12/12/14.

[100] D. Sabatier. Using Formal Proof and B Method at System Level for Industrial Projects. Proceedings of Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification - First International Conference, RSSRail, Lecture Notes in Computer Science 9707, Springer, ISBN 978-3-319-33950-4, Paris, France, June 28-30, 2016.

[101] BS EN50126: Railway applications: The specification and demonstration of Reliability. Availability, Maintainability and Safety (RAMS), 1999.

[102] BS EN50129: Railway applications Communication, signalling and processing systems Safety related electronic systems for signalling, 2003.

[103] BS EN50128: Railway applications - Communication, signalling and processing systems Software for railway control and protection systems, 2011.

[104] RTCA DO-178C (EUROCAE ED-12C), November 2011, Software Considerations in Airborne Systems and Equipment Certification.

[105] RTCA DO-178B (EUROCAE ED-12B), 1992, Software Considerations in Airborne Systems and Equipment Certification,Washington DC.

[106] ARP4754A, 2010, Guidelines for Development of Civil Aircraft and Systems, SAE International.

[107] ARP4761, 1996, Guidlines and Methods for Conducting the Safety Assessment process on Civil Airbone Systems And Equipement.

[108] ISO 26262, Road vehicles Functional safety. International Standard, November 2011.

[109] IEC61508: Functional safety of electrical/electronic/programmable electronic safety-related systems, 2010.

[110] DO-254, 2000, Design Assurance Guidance for Airborne Electronic Hardware, RTCA.

[111] DO-278/ ED-109 Guidelines for communication, navigation, surveillance and air traffic management systems software integrity assurance, RTCA/EUROCAE, March 2002.

[112] DO-297/ED-124 Integrated Modular Avionics (IMA) Development guidance and certication considerations, RTCA/EUROCAE, December 2010.

[113] G. da Cunha Trivelato, M. L. de Oliveira e Souza. Current Trends Driving Aircraft and Automotive Systems Architectures and their impacts onCMMI® Organizational Structures. http://www.aviation-conferences.com/architecture/pdf/paper-gilberto.pdf.

[114] U.K. Ministry of Defence, 00-56 Safety Management Requirements for Defence Systems, Ministry of Defence, Defence Standard, December 1996.

[115] Gorski J (2004) Trust Case – a case for trustworthiness of it infrastructures. In Proc. NATO Advanced Research Workshop on Cyberspace Security and Defence: Research Issues, Gdansk, Poland.
[116] R. Bloomfield and P. Bishop. Safety and Assurance Cases: Past, Present and Possible Future - an Adelard Perspective. Making Systems Safer. ISBN 978-1-84996-085-4. Springer-Verlag London, 2010, p. 51, DOI 10.1007/978-1-84996-086-1_4.
[117] SACM: http://www.omg.org/spec/sacm/1.1.
[118] GSN: Community Standard Version 1, 2011.
[119] F. Redmill. Analysis of the COTS Debate. Safety Science 42 :355-367.
[120] SACM: http://www.omg.org/spec/sacm/1.0.
[121] E. Denney and S. Trac. A Software Safety Certification Tool for Automatically Generated Guidance, Navigation and Control Code. In IEEE Aerospace Conference, Big Sky, MT, USA, 2008.
[122] SEI, Carnegie Mellon. What software product line are not. http://www.sei.cmu.edu/productlines/frame_report/pl_is_not.htm. [Accessed 18-09-2016]
[123] P. Clements, L. Northrop. *Software Product Lines: Practices and Patterns*. Boston, MA: Addison-Wesley, 2002.
[124] S Bates et al. Safety Case Architectures to Complement a Contract-Based Approach to Designing Safe Systems. Proceedings of 21st International System Safety Conference, 2003.
[125] ISO 26262:2018 https://www.iso.org/obp/ui/#iso:std:iso:26262:-2:dis:ed-2:v1:en
[126] ISO-SAE 21434 Road vehicles –Cybersecurity Engineering- General Overview. https://www.iso.org/standard/70918.html
[127] Young, A., & Walker, A. (2017). Improvements in Functional Safety of Automotive IP Through ISO 26262:2018 Part 11. In Proceedings of 24th European Conference on Systems, Software and Services Process Improvement (EuroSPI), Ostrava, Czech Republic, September 6–8, 2017, pp. 547-556, Cham Springer International Publishing. https://doi.org/10.1007/978-3-319-64218-5_45.
[128] DO-330, Software Tool Qualification Considerations, RTCA & EUROCAE, 2011.
[129] European Cooperation for Space Standardization ECSS-Q-ST-30C Space product assurance - Dependability 06/03/2009
[130] European Cooperation for Space Standardization ECSS-Q-ST-40C Space product assurance - Safety 06/03/2009
[131] Secure Software Engineering Standard ESSB-ST-E-008 04/07/2016.
[132] AMASS project: D5.1 Baseline and Requirements for Seamless Interoperability. http://amass-ecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D5.1_Baseline-and-Requirements-for-Seamless-Interoperability_AMASS_Final.pdf, 2016.
[133] P. Baufreton et al., “Multi-domain comparison of safety standards”, 1st European Congress Embedded Real Time Software and Systems (ERTS)-2010.
[134] J. Machrouh et al., “Cross domain comparison of System Assurance”, 3rd European Congress Embedded Real Time Software and Systems (ERTS)-2012.
[135] E. Ledinot et al., “A cross-domain comparison of software development assurance standards”, 3rd European Congress Embedded Real Time Software and Systems (ERTS)-2012.
[136] JP. Blanquart et al., “Criticality categories across safety standards in different domains”, 3rd European Congress Embedded Real Time Software and Systems (ERTS)-2012.
[137] E. Ledinot et al., “Joint use of static and dynamic software verification techniques: a cross domain view in safety critical system industries”, 5th European Congress Embedded Real Time Software and Systems (ERTS)-2014.
[138] JL Camus. Tool Qualification in Multiple Domains: Status and Perspectives. 5th European Congress Embedded Real Time Software and Systems (ERTS)-2014.
[139] JP. Blanquart et al., Software Safety-A journey across domains and safety standards, 9th European Congress Embedded Real Time Software and Systems (ERTS)-2018.