A Harmonized Compositional Assurance Approach for Safety-Critical Systems

Tesis doctoral presentada por B. Alejandra Ruiz López
dentro del Programa de Doctorado en Informática y Telecomunicaciones
Dirigida por Dr. Huáscar Espinoza Ortiz
y Dr. Prof. Tim P. Kelly

UNIVERSIDAD DE DEUSTO

A mi madre

To my mother

Abstract

Safety-critical systems, those whose failure could end up in loss or injuries to people or the environment, are required to go through laborious and expensive certification processes. These systems have also increased their complexity and as it has already been done in other domains, they have applied component-based system developments to deal with complexity. However, components are difficult to assess as certification is done at system level and not at component level. Compositional certification approach proposes to get incremental credit by accepting that a specific component complies with specific standard's requirements and it is correctly integrated. The objective is to support integration of new components while the previously integrated components do not need to work for re-acceptance.

We propose **(1) the use of assurance modelling techniques** to provide us the mechanism to understand the common basis of standards shared by different domains such as the avionics, automotive and the medical devices design.

We propose **(2) an assurance decomposition methodology** offering guidance and modelling mechanisms to decompose the responsibilities associated with the life-cycle of safety-critical components. This methodology ensures a hierarchy of assurance and certification projects where the responsibilities and project tasks can be specified and its accomplishment can be assessed to determine the compliance of functional safety standards.

Assurance decomposition supports the reuse of components as it guides us not just for standards compliance but specifically on the understanding and tailoring of those standards for component assurance and support when those components are integrated into the final system.

We propose **(3) a contract-based approach** to support the integration of reused components and at the same time, the proposal supports the identification of assumptions, a very laborious and time consuming task. Assurance Contracts are defined to ensure incremental compliance once the components are integrated. The objective of this assurance contracts is to ensure the overall compliance of the system with the selected standards and reference documents such as guidelines or advisory circulars.

The defined approach to assurance contracts specification attempts to balance the need for unambiguity on the composition while maintaining the heterogeneity of the information managed. The claims classification offers an easy method to support the assessment of contract completeness and the structured expressions provide a semi-formal language to specify the assumptions and guarantees of a component.

This work has been mainly framed in a European collaborative research projects such as OPENCOSS a Large-scale integrating project (IP) with 17 partners from 9 countries to develop a platform for safety assurance and certification of safety-critical systems

(compliance with standards, robust argumentation, evidence management, process transparency), SAFEADAPT an FP7 project with 9 partners and RECOMP an ARTEMIS project..

The results of this work have been presented to the standardization group of the Object Management Group responsible for the SACM (Structured Assurance Case Metamodel) standard specification, which currently discusses its inclusion in future versions.

The **(4) tools** presented and used in this work have been included in the results of an open tool platform developed within the OPENCOSS project that is being released in PolarSys. PolarSys is an Eclipse Industry Working Group created by large industry players and by tools providers to collaborate on the creation and support of Open Source tools for the development of embedded systems.

Resumen

Los sistemas de seguridad críticos, aquellos que en caso de fallo pueden inducir la pérdida o el daño de personas o el entorno, están obligados a cumplir con minuciosos y costosos procesos de certificación. Los sistemas críticos al igual que se ha hecho en otros dominios, han aplicado diseños de sistemas basados en componentes para hacer frente a la complejidad. Sin embargo, el cumplimiento de requisitos de una certificación dada es difícil de evaluar en los componentes. Esta evaluación se realiza a nivel del sistema y no a nivel de componente. El enfoque de certificación composicional propone obtener créditos intermedios al aceptar que un componente específico cumple con los requisitos de un estándar seleccionado. El objetivo es apoyar la integración de nuevos componentes a la vez que se mantiene a los componentes existentes ya integrados sin la necesidad de re-aceptación.

Se propone **(1) el uso de técnicas de modelado de aseguramiento** que nos proporcione el modo de comprender sin ambigüedad de manera común por diferentes dominios tales las de aviónica, automoción y el diseño de dispositivos médicos, la base de normas de seguridad.

Se propone **(2) una metodología de descomposición del aseguramiento del sistema** que ofrece guías y mecanismos de modelización para descomponer las responsabilidades asociadas con el ciclo de vida de los componentes críticos para la seguridad. Esta metodología garantiza una jerarquía de proyectos de aseguramiento y certificación, donde las responsabilidades y tareas del proyecto se pueden especificar y su realización se puede evaluar para determinar el nivel de cumplimiento de las normas de seguridad funcional.

La descomposición de aseguramiento soporta la reutilización de componentes, ya que ofrece una guía no sólo para el cumplimiento de las normas, sino también específicamente para la comprensión y la adaptación de los criterios de garantía de los componentes y apoyo cuando esos componentes se integran en el sistema final.

El **(3) enfoque basado en contratos** propuesto apoya la integración de componentes reutilizados y, al mismo tiempo apoya la identificación de asunciones, una tarea muy laboriosa y que consume mucho tiempo. Los contratos de aseguramiento se definen para asegurar el cumplimiento gradual en el momento de integración de los componentes. El objetivo de estos contratos de aseguramiento es garantizar la conformidad global del sistema con los estándares seleccionados y documentos de referencia tales como directrices o circulares de asesoramiento.

El enfoque definido con la especificación contratos de aseguramiento intenta equilibrar la necesidad de reducir ambigüedad sobre la composición mientras se mantiene la heterogeneidad de la información gestionada. La clasificación de las afirmaciones ofrece un método sencillo para apoyar la evaluación de la integridad del contrato y las

expresiones estructuradas proporcionar un lenguaje semi-formal para especificar las asunciones y garantías del contrato.

Este trabajo se ha enmarcado dentro de proyectos europeos de investigación en colaboración, principalmente OPENCOSS, un proyecto de gran escala (IP) con 17 socios de 9 países para desarrollar una plataforma para la garantía de la seguridad y la certificación de los sistemas críticos de seguridad (cumplimiento de las normas, argumentación robusta, gestión evidencias, proceso transparentes), SAFEADAPT un Proyecto del 7º programa marco con 9 socios y RECOMP, un Proyecto ARTEMIS.

Los resultados de este trabajo se han presentado al grupo de estandarización de la Object Management Group responsable de la especificación del estándar SACM (Structured Assurance Case Metamodel) y se está en trámites de discusión para su inclusión en futuras versiones.

Las **(4) herramientas** presentadas y utilizadas en este trabajo se han incluido en los resultados de una plataforma de herramientas abierta desarrollada dentro del proyecto OPENCOSS que se ha liberado en PolarSys. PolarSys es un grupo industrial de trabajo de Eclipse creado por representantes de grandes industrias de sistemas embebidos y proveedores de herramientas para colaborar en la creación y el apoyo para el desarrollo de herramientas de código abierto.

Acknowledgements

This Ph.D. will not have been possible without the support of many people. The first one to thank is my mother. She has been there from the beginning when I came home one day wondering if it would be a good idea to work on a Ph.D. I remember her answer: Alejandra, you can do it! She has been monitoring the progress from the very beginning until the end; she knew when I was writing a paper for a conference, when I was working on a chapter and when I was stuck on a concept. Every time she saw me desperate she has told me, “let’s have and ice-cream and talk”. The other pillar of this thesis has been Huascar my supervisor. He has been the first one to propose me to work on critical-systems and make me improve by his endless discussion full of questions. He was the one that suggest me I could work on a Ph.D. on this topic. He has been supporting me all the time not only being my supervisor but also a good friend. I would also like to thanks Tim Kelly my other supervisor for sharing with me so good discussions on safety. It has been a pleasure to work together and I am looking forward to continue working together on this interesting topic which is safety.

Thank you Paulo Barbosa and your team at NUTES, your inputs and feedback on this thesis have been incredible. I would like to mention Alberto Melzi from Centre Riserche di Fiat for those long discussions on the SEooC (Safety Element Out Of Context) and the people from the Thales airworthiness directorate, Cedric Chevrel, Frank Aime and Cyril Marchand who introduce me on the complex world of avionics standards compliance. This work will not be the same without Philippa Conmy who discusses with me about modular argumentation and Katrina Attwood for her support on language structures or Paolo Panaroni for his enthusiasm and interesting comments of approaches and comparison from similar projects and approaches. I would like to specially mention Jose Luis de la Vara not only for his technical support on modelling but also for becoming a friend during the process.

I need to mention the people from Fraunhofer-ESK, especially Gereon, Philipp and Dulcineia thanks for sharing your expertise with me. Also people from LISE lab at CEA, Nataliya who shared her expertise in safety analysis and system modelling or Mauricio and Juan Jose who not only help me during my stay at CEA but have become good friends and make me feel at home.

I only have good words to say for the team at Tecnalia who shared with me this process, Mª Carmen, Idoya, Angel, the tool support presented on this thesis would not be possible without your support. Garazi, Maite, Jason, Xabier, Sergio and Ana thanks for your support in everything during this process. Your collaborations technically, logistically and your good mood has make this possible.

I would like to also thank my family and friends for their support during all this time. Thanks to my uncle Alvaro to make the Sunday’s family lunches a thesis progress report

and finally last but not least thank you so much to Iñigo for do not letting me became mad in the process.

I will probably forget someone, please forgive me. I believe this thesis is also yours.

Agradecimientos

Este doctorado no hubiera sido posible sin el apoyo de muchas personas. La primera a darles las gracias es a mi madre. Ella ha estado ahí desde el principio, cuando llegué a casa un día preguntándome si sería una buena idea eso de hacer un doctorado. Recuerdo su respuesta: Alejandra, ¡tú puedes hacerlo! Ella ha estado siguiendo el progreso desde el principio hasta el final; ella sabía cuándo estaba escribiendo un artículo para una conferencia, cuándo estaba trabajando en un capítulo y cuándo me he quedado atascada en un concepto. Cada vez que me ha visto desesperada que me ha dicho, "vamos a por un helado y me cuentas". El otro pilar de esta tesis ha sido Huáscar, mi supervisor. Él fue el primero que me propuso trabajar en sistemas críticos y me ha hecho mejorar a base de sus interminables discusiones llenas de preguntas. Fue él quien me sugirió que podría trabajar en un doctorado en este tema. Él me ha estado apoyando todo el tiempo, no sólo siendo mi supervisor, sino también un buen amigo. También me gustaría agradecer a Tim Kelly mi otro supervisor por compartir conmigo tan buenas discusiones sobre seguridad. Ha sido un placer trabajar juntos y estoy con ganas de volver a trabajar juntos en este tema tan interesante.

Gracias Paulo Barbosa y su equipo en NUTES, sus contribuciones y comentarios en esta tesis han sido increíbles. Me gustaría mencionar Alberto Melzi del centro Riserche di Fiat por esas largas discusiones sobre el SEooC (Safety Element Out Of Context) y las personas del equico de airworthiness directorate de Thales, Cedric Chevrel , Frank Aime y Cyril Marchand que me introdujeron en el complejo mundo de las normas de aviónica. Este trabajo no sería el mismo sin Philippa Conmy y sus discusiones sobre la argumentación modular, Katrina Attwood por su apoyo en las estructuras del lenguaje o Paolo Panaroni por su entusiasmo y comentarios interesantes de enfoques y de comparaciones con otros proyectos y enfoques similares. Me gustaría mencionar especialmente a José Luis de la Vara no sólo por su apoyo técnico en el modelado, sino también por convertirse en un amigo durante el proceso.

Debo mencionar con cariño a las personas de Fraunhofer - ESK, especialmente Gereon, Philipp y Dulcineia gracias por compartir vuestra experiencia y conocimientos conmigo. También la gente del laboratorio LISE en el CEA, Nataliya que compartió sus conocimientos en análisis de seguridad y el modelado de sistemas o Mauricio y Juan José, que no sólo me ayudaron durante mi estancia en el CEA, sino que además se convirtieron en buenos amigos y me hicieron sentir como en casa.

Sólo tengo buenas palabras para el equipo de Tecnalia que ha compartido conmigo este proceso, Mª Carmen, Idoya, Ángel, el desarrollo de las herramientas que se muestra en la tesis no sería posible sin vuestro apoyo. Garazi, Maite, Jason, Xabier, Sergio y Ana gracias por vuestro apoyo en todo durante este proceso. Vuestra ayuda técnica, logística y buen humor han hecho que esto sea posible.

Me gustaría dar las gracias también a mi familia y amigos por su apoyo durante todo este tiempo. Gracias a mi tío Álvaro por convertir las comidas familiares del domingo en un informe de avance de la tesis y, finalmente, pero no por ello menos importante gracias a Iñigo por no dejarme que me volviera loca en el proceso.

Probablemente olvide a alguien, por favor, perdonadme. Creo que esta tesis es también vuestra.

Contents

List of figures

List of tables

"Concern for man himself and his safety must always for the chief interest of all technical endeavours." -Albert Einstein

1

Introduction

1.1 Introduction

Safety-critical systems are defined as computer-based systems that in case of an incident or misbehaviour can lead to an accident that will put people or the environment in danger, resulting in injuries and or casualties [Knight 2002]. To deal with this issue even further, different standards and guidelines for the design of safety-critical systems have been created, such as the IEC 61508 standard for Functional Safety[1] of Electrical/Electronic/Programmable Electronic Safety-related Systems [IEC 61508]. One of the main challenges for those standards is that they should not avoid technological innovation but at the same time, the systems resulting from those innovations should remain safe. In order to cope with that, those standards and guidelines tend to be sometimes ambiguous and open to different interpretations. While those interpretations leave the door open to new ideas, technologies and methods, they also make it difficult for standard compliance assessors and companies to share the same views. New technologies need to prove safety prior to their inclusion in safety-critical systems.

Safety-critical systems have increased in technical complexity towards open, upgradeable and interconnected systems, exacerbating the problem of ensuring safety in the presence of human, environmental and technological risks. These systems also need to handle with a shorter and shorter time to market. This situation is pulling companies to invest in reusing parts or whole systems from one project to another, sometimes even from completely different domains. Although this solution seams very reasonably at first it is not so easy to implement and put into practice on safety-critical domains and new problems emerge. Safety is a system property and it is not easy to define its boundaries while trying to decompose it among the constituent parts and components that interact for its operation. Assurance[2] of safety-critical systems is done at high level, not at

[1] Functional Safety is the part of the overall safety of a system or piece of equipment that depends on the system or equipment operating correctly in response to its inputs, including the safe management of likely operator errors, hardware failures and environmental changes. [IEC 61508].

[2] System Assurance is the planned and systematic activities that assure systems engineering processes and products conform with its requirements for safety, reliability, availability, maintainability, standards, procedures, and regulations.

component level. So when dealing with activities such as its certification[3], their correctness assessment is done by composing small parts into the whole system. This assessment must assure that the system integration is done correctly and that new properties or behaviours will not emerge unnoticed on the integration.

This thesis work focuses on three industrial domains: avionics, automotive and medical devices as an exemplary subset. All these domains share two aspects that makes them relevant for this work; they are all involve safety-critical areas and their systems catalogue are getting more and more complex and dependent on software. We will specifically analyse how components reuse is being handled in relation to assuring compliance with safety standards. This thesis deals with the management of certification assets in order to support the reuse of parts of the system or the whole system within new projects. The underlying work also deals with modelling requirements from the safety standards in relation with reuse, and how the assurance information for a component is managed along the lifecycle and used during composition. When the component is reused and integrated into the system, assurance of safety standards compliance is addressed by composing assurance assets from each of the components that form the whole systems. This compositional approach could support the assurance of complex systems in very regulated and standardized areas such as the avionics domain and be flexible to incorporate new technological trends.

The rest of this chapter is organized as follows. Section 1.2 explains the purpose of this work. Section 1.3 details the problem that the present thesis resolves. Section 1.4 introduces the goals defined for this work. Section 1.5 describes the approach followed in this thesis to fulfil the identified goals. Section 1.6 introduces the research methodology that has been followed in this work. Section 1.7 explains the context in which the work of this thesis has been performed. Finally, Section 1.8 gives an overview of the structure of this document.

1.2 Motivation

The society is requesting more complex systems and with shorter lead development times to bring those systems into the market. In order to reduce the time requested to put a new system into the market, reuse of software improve productivity, reliability and lower overall cost in software development projects. This improvement could increase up to 50% with high level of reuse [Gill 2003].

Component-based development increases the potential for reuse and it is being used in safety-critical systems as a response to the increasing complexity in its design and as a reusability mechanism. Modern engineering and business practices use massive

[3] Certification is a (legal) recognition that a system complies with standards, rules and regulations designed to ensure it can be depended upon to deliver its intended service safely.

subcontracting supply chains and Commercial Off The Shelf (COTS)[4] component-based development.

However, current safety certification schemes are often criticised for being process-centric, focusing on the development process and less on the resulting product. This has a negative impact on component-based (or systems of systems[5]) environments where there can be little and/or poor visibility into the internal subsystems design process.

In the avionics domain, experience shows that despite the difficulties and costs incurred over the certification of COTS components, these components pose relatively few problems, and in most cases, with only minor negative impact. This observation suggests that the required levels of safety can be met by adopting broadly-used COTS products, thus laying the groundwork for a reuse strategy in aerospace system design [Espinoza et al. 2011].

Functional safety standards such as DO-297 [DO-297] in the avionics domain are introducing concepts for components reuse and moving from federated systems towards a component-based and modular systems architecture. Safety assurance demands a systems perspective where software and/or hardware components can be integrated providing the desired effect.

In the automotive domain, the ISO 26262 standard for functional safety [ISO 26262] has introduced the concept of SEooC (Safety Element out of Context) where a component is evaluated against a "presumed" operational context and operating conditions. Once the component becomes part of a specific system in an actual operational context, the evaluation is completed by comparing assumed context conditions against actual context conditions.

In the medical domain, devices shall also follow functional safety standards such as the IEC 62304 [IEC 62304] for software developments. The concept of SOUP (Software of software of unknown provenance) is introduced as a concept to apply when dealing with COTS. One of the main differences with other domains is that the evidence to provide is not so clearly specified and it is the manufacturer the responsible to specify them.

However, the functional safety standards lack details regarding the adoption of a compositional approach. This absence of guidelines is noticed in all safety-critical related systems. There is a need to tackle the composable/modular system view of the certification problem. This would imply that (a) certification approaches should be extended to detail certification data in terms of the component/system and, (b) they must address technology, policy and personnel involved in a component-based system.

[4] Commercial-Off-The-Shelf (COTS) software – Commercially available applications sold by vendors through public catalogue listings. COTS software is not intended to be customized or enhanced. Contract-negotiated software developed for a specific application is not COTS software.[DO-178c]

[5] Systems of systems are large scale concurrent and distributed systems that are comprised of complex systems.[Kotov 1997]

The responsibilities for certification should be properly understood and accepted by the different stakeholders involved in the system development and assessment. When the system development is decomposed across different suppliers these suppliers also inherit some of these responsibilities and should be aware of the implications on the whole system assurance.

The main prerequisite for achieving the level of transparency expected for safety-critical systems is therefore to answer the following questions: (1) what information is required by each stakeholder to achieve the required level of transparency and trust?, (2) what is the best way to represent such information contained in existing standards, practices, and technologies?

1.3 Problem statement

Different projects have conducted different surveys in order to capture the challenges and state of the art on these areas. We will particularly highlight the outcomes of the OPENCOSS project.

Partners from the OPENCOSS project [OPENCOSS D1.1] have claimed that the expected benefits that can be achieved by reusing components, may lead to development efficiency by time saving of and cost reduction, due to reduction of the effort of creating the software, testing, debugging and fine-tuning. Such reuse of components can make certification more systematic and manageable, due to better analysis of system integration issues (emergent issues[6]). In some domains such as IT systems, composition of systems from existing certified generic products results in reduction of the re-certification costs and efforts for reused parts.

However, in order to achieve such benefits as described above, a set of challenges should be tackled. Certification and safety assurance can be error prone. It is difficult to justify the safety of a reused component without providing a full certification dossier. There is the risk of different contexts jeopardising implicit assumptions made about compatibility which compromise system safety. In the automotive sector a new concept SEooC has been introduced. The SEooC shall be evaluated against “presumed” operational context conditions. When reused components are not integrated in the overall system development lifecycle, the gap at the boundary between system requirements and generic product features may lead to risks of discovering system integration constraints. In case of the SEooC, it shall be evaluated by comparing assumed context conditions against actual context condition. Moreover, a component certification data should be complete and consistent, containing consistent evidence management, component models and specifications, and data in order to assess the possibilities for reuse in another context. Component-level arguments should be encapsulated effectively.

[6] Emergent behaviour is that which cannot be predicted through analysis at any level simpler than that of the system as a whole [Dyson George 1997].

1.3.1 State of the practice

Compositional assurance deals with component-based development. The reason for decomposing the system into components is in order to benefit from the possible reuse of the components. In a similar way, compositional assurance search to benefit from reuse of already assurance component and reduce the overall effort for system assurance. We face a reuse challenge related to how reuse is managed, what is the scope of reuse and what reuse methodology we must apply. For a better clarification of the problem please refer to Fig. 1, a reuse taxonomy created to support the reader.

Each of the branches in the diagram shows a different type of reuse. There are scenarios that combine different reuse types depending on the perspective. In Fig. 1 some boxes are highlighted in green; this has been done in order to define the scope of this research.

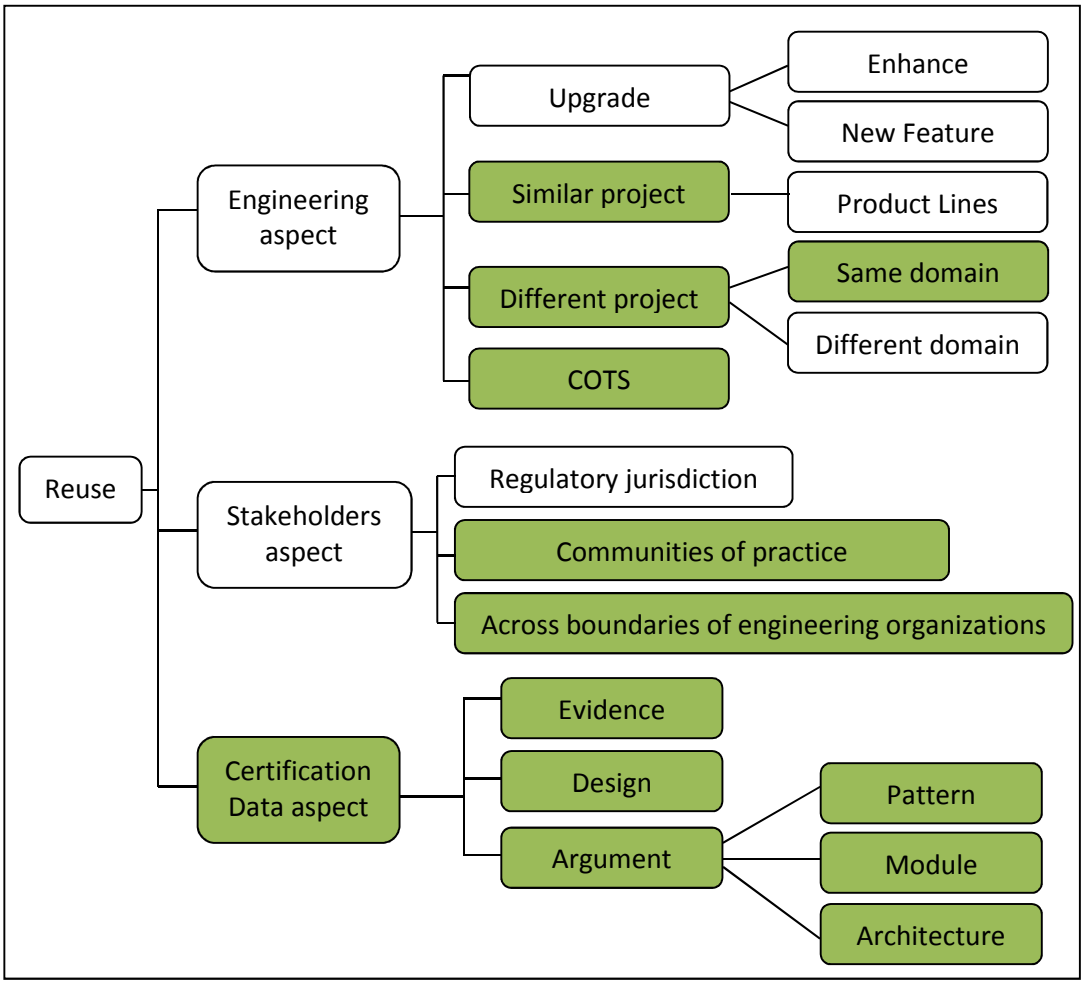


Fig. 1 Taxonomy reuse

Engineering aspect

This aspect focuses on the idea of reuse on 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 aspects or mechanism associated with the engineering decisions are reused. However, the reuse of work products related with demonstration compliance with standards which are done together with the component (HW and/or SW) development are usually impacted and need some rework. The reuse scenarios are the following:

- **Upgrade – new feature**. A component which is associated with a particular hardware and/or software will include new features that previous component did not have. We have a basic component that will include a new feature in the next version.
- **Upgrade – Enhance performance**. A component which is 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 the performance will be enhanced.
- **Similar project**. A component is reused and integrated into a system with the same context and domain as the previous used. The functionalities needed in both projects are the same and so the component is reused.
- **Similar Project - Product Lines**. According to the Software Engineering Institute (SEI) description, a software product line 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 that are developed from a common set of assets in a prescribed way [Clements Northrop 2001].
- **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 are from the same domain. The operational environments and/or systems in which the component is integrated might differ.
- **Different project –different domain**. General use components may be reused not only in projects from the same domain but also in/from project from different domains.
- **COTS.** In this case, the reuse is of a component described as “Commercial off the Shelf”. These components are usually general purpose components that can apply to different domains and purposes.

In this work we will focus specifically on reuse of **similar and different projects in the same domain** and the reuse of **COTS**. The other aspects, although relevant, are outside the scope of this research.

Stakeholders aspect

This aspect focuses on the different interests depending on the stakeholders. The objective here is not so much led 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 on different countries. In this case different jurisdictions apply to the same product/ component and the certification artefacts generated for one jurisdiction can be applied in order to achieve other jurisdiction. When components are used on different countries, the different jurisdiction of each of the countries shall be taken into account in their design
- **Communities of practice**. Reuse of the methodologies and practices of one or more activities mentioned on the standard between different communities who share the same objectives
- **Across boundaries of engineering organizations**. Different groups within the same organization or across different organization reuse the components (designs, implementation and/or certification artefacts) from one group to another.

In this work the reuse between communities of practice and across boundaries of engineering organizations are in the scope of this research.

Certification data aspect

This aspect is focused on the reuse of certification related data and work products. The objective here is to reuse the data generated on previous projects or phases and minimise the rework of certification related activities.

- **Evidence**. Reuse of the results from conducting a 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.
- **Design**. Reuse of the results from the design phase into different projects.
- **Argumentation – Patterns**. Safety case patterns are considered to be one of the main approaches for managing reuse of safety assurance. A safety case pattern provides a means of explicitly and clearly documenting common elements found in safety cases, and it also promotes the reuse of best practices for safety assurance [Hawkins Kelly 2013].
- **Module - Safety Case**. Safety cases modules are parts of an overall safety case containing part of an argument and relevant citations of evidence. A safety case module may correspond, among other things, to an interrelated set of safety engineering activities, scope of responsibilities of a particular engineering organisation, well-defined sub-system or equipment used within the overall safety-critical platform [OPENCOSS D5.2].
- **Argumentation Architecture.** Safety case architecture is defined by Kelly 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 exit between them” [Kelly 2003]. Argumentation architecture supports the reuse of argument modules which can be associated to system component.

The focus of this work is all the above scenarios.

1.3.2 Problem synthesis

Compositional assurance is considered a system property and previous works have been focused on safety property decomposition across the system elements but not on the work required to reuse the certification-related data when the elements of the system are reused. The challenge developers and practitioners face is the lack of guidelines and support for component-based development and how to take advantage of reuse mechanism while ensuring compliance with standards. In particular, the challenges that this thesis addresses can be stated by the following three research questions in the assurance and certification of safety-critical systems:

Research question 1 (RQ1). How can we express standards compliance needs in relation to component development and component integration?

Research question 2 (RQ2). How can we express the distribution of responsibility across the system stakeholders in relation to safety standards compliance?

Research question 3 (RQ3). How can we best support stakeholders while ensuring compliance across the system development lifecycle?

These research questions are analysed and answered in the following sections.

1.4 Thesis goals

At section 1.3 of this thesis we have described the main problems safety-critical systems assessors face when assuring safety standards compliance for a component-based system. In chapter 1 we defined the scope of this thesis on three perspectives to improve:

- **Perspective 1: Guidance**. It consists of providing guides, formalisation of certification knowledge, and methodology. It is a way to structure the knowledge base and explore that knowledge to be used by different stakeholders.
- **Perspective 2: Reuse parts/components**. This perspective is focused on managing the decomposition of responsibilities of the work at component level and integration at system level.
- **Perspective 3: Automation**. The objective here is the automation time consuming aspects of compositional assurance. We aim in providing support to the impact analysis thought the right identification of use context and parameters at system, hardware and software level. A methodology and tools to support integration of components into a system from the assurance perspective is provided.

The aim of this work is to provide an answer to the research questions presented above at the different levels of abstraction. Next, the main contributions of this research work are summarized.

Thesis answer to RQ1 is to elaborate some guidelines and a mechanism to express in a clear way which information is requested by the standards for components to be reused in safety-critical. The aim is to provide a methodology that will serve both developers and

practitioners on the understanding on the requirements for the components safety compliance and reuse.

Thesis answer to RQ2 is the decomposition of responsibilities and work developed by each of elements in which the system is decomposed. When decomposing a system into components and reusing components, the responsibilities for assurance can be ambiguous and the responsibilities of each of the stakeholders may be unclear.

Thesis answer to RQ3 is based on supporting the stakeholders on the assurance process, providing the status information regarding the component and system assurance.

1.5 Thesis approach

This work is based on the Common Certification Language (CCL) developed in the OPENCOSS project [OPENCOSS D4.4]. This is a model-based approach for the specification of safety compliance needs for critical systems. The approach is based on a holistic and generic metamodel that 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, which correspond to a model of the safety criteria of a given standard. The metamodels have been validated versus safety standards and with practitioners. The metamodels support the specification of safety compliance needs for most critical computer-based and software-intensive systems.

Associated with component-based development, contract-based approaches have been developed in order to help address component integration. However, contract-based approaches differ when we see them from the development perspective and from the safety assurance perspective. This thesis is focused on *assurance contracts* which are defined as a set of claims that need to be made concerning a component to support its certification against a particular safety assurance standard.

Specifically, our approach provides the following contributions:

The Assurance modelling contribution. The common certification language (CCL) is applied in the scope of component-based system assurance. The interpretation of the CCL metamodels proposed in this work provides the mechanism to understand objectives of standards with a common basis. This is the first step to generate guidelines and formalize the compliance with standards for a company and a project. Modifications proposed to the Structured Assurance Case Metamodel (SACM)[7], are included along with the CCL for the component development and reuse. This contribution is described in detail in chapter 4.

The Compositional assurance decomposition contribution can be described as a contribution for perspectives 1 and 2. It provides us with a guideline and the mechanism to decompose the responsibilities associated with component with the objectives of

[7] The Structured Assurance Case Metamodel is an OMG (Object Management Group) standard developed by the SysA (System Assurance) group.

standards based assurance. We are able to identify a hierarchy of assurance projects where the responsibilities and tasks can be specified and there is a mechanism to indicate compliance of those tasks. This contribution is described in detail in chapter 5.

The Contract-based approach for assurance integration contribution tries to address perspective 2 and perspective 3. A contract-based approach is defined to support the integration of reused components. The proposal supports the identification of assumptions currently a very laborious and time consuming task. This contribution is described in detail in chapter 6.

In order to fulfil perspective 3 of the scope and support the previous contributions a **tool support contribution** has been developed and applied on the different case studies. The tool implements the metamodels shown on the assurance modelling chapter and provides editors for each of them. The tool support is not explained in a specific chapter but has been shown on the different chapters and been used on the case studies. Tool support contribution is described in chapters 4, 5 and 6.

1.6 Research methodology

In order to perform the work of this thesis, we have carried out a research project following the case study research methodology defined by Yin [Yin 2013]. According to [Yin 2013] a case study is the preferred method when '(a) "how" or "why" questions are being posed, (b) the investigator has little control over events, and (c) the focus is on a contemporary phenomenon within a real-life context. This is the case of this study and the research questions identified previously can be categorized as "how" questions.

We have adapted the iterative case study design process proposed by Yin and conducted three case studies using the following phases (See Fig. 2):

- **Phase 1:** it focuses on defining the research questions, validating the rationale behind choosing a case study as the preferred method; develop a theory by a deep analysis of the state of the art.
- **Phase 2** works in parallel executing the three case studies. The case study development cycle consists of 5 process steps: (1) design the case study by preparing the procedures, (2) prepare to conduct the case study pilot, (3) collect the evidences for the case study, (4) analyse the case study evidence in contrast with the proposed theory , (5) share the case study report
- **Phase 3:** collects the results for each of the case studies for common analysis and produce common conclusions.

In order to produce the case study procedures and reports the guidelines from Runeson and Höst [Runeson Höst 2009] have been followed. They proposed some checklists to ensure the quality of the case study design and reports. We have been using those checklists in order to verify the quality of the case studies proposed in this work.

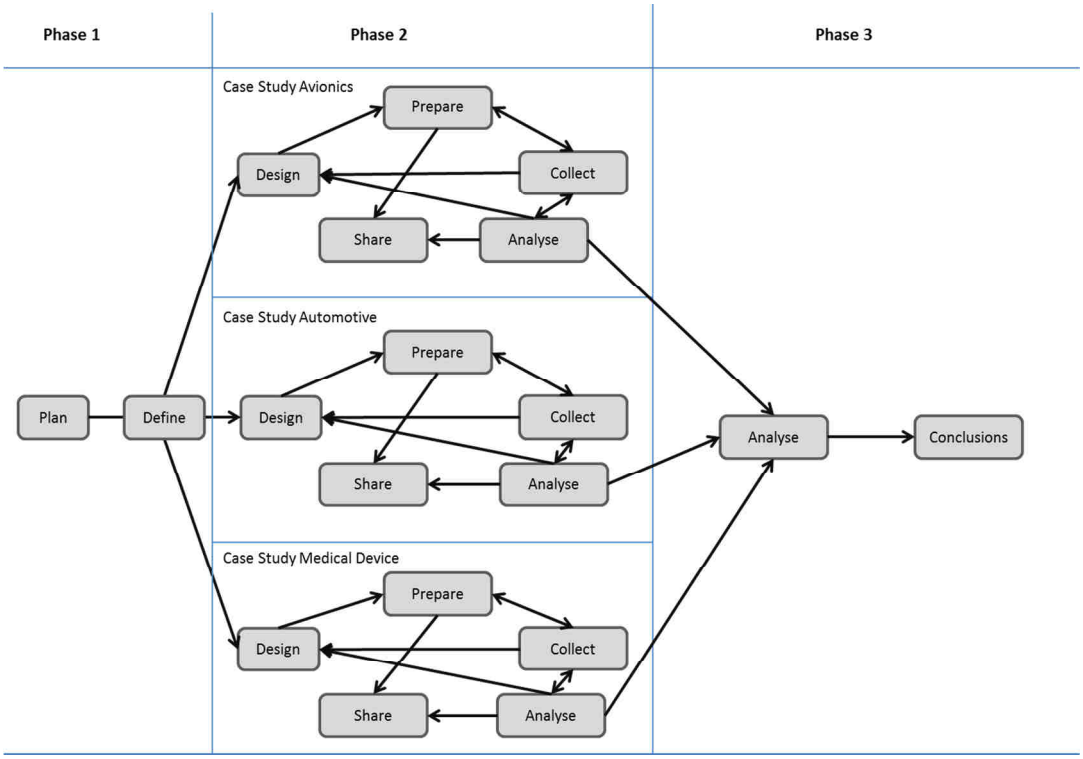


Fig. 2 Case Study research process based upon[Yin 2013]

1.7 Thesis context

This thesis has been developed in the context of the research center Fundación TECNALIA Research & Innovation [TECNALIA]. The work that has made the development of this thesis possible is in the context of the following research funded projects:

- RECOMP: "RECOMP" stands for Reduced Certification Costs Using Trusted Multi-core Platforms and is a European funded project from ARTEMIS JOINT UNDERTAKING (JU). The project started April 1th of 2010 and has duration of 36 months.
- OPENCOSS: OPENCOSS is a European large scale integrating FP7 project dedicated to produce the first European-wide open safety certification platform: an Open Platform for EvolutioNary Certification Of Safety-critical Systems for the railway, avionics and automotive markets.
- SAFEADAPT: Safe Adaptive Software for Fully Electric Vehicles a European FP7 project, is analyzing adaptation to address the needs of full electric vehicles regarding safety, reliability and cost-efficiency. This project is planned to finish in March 2016.

1.8 Thesis outline

This thesis is presented in nine chapters including this one, and one appendix. The remainder of this thesis is organized as follows:

Chapter 2 introduces the main fields that are related to the work that is presented in this thesis in order to provide the reader with the background knowledge and research context for the thesis work.

Chapter 3 presents an explanation on the assurance standard ecosystems and its application in the three domains under study: avionics, automotive and medical devices. The analysis focuses on the composition problem and how component reuse is handled in assurance standards.

Chapter 4 introduces model driven compliance engineering by the appliance of the Common Certification Language to the composition problem.

Chapter 5 proposes a hierarchical decomposition of the assurance problem aligned to the system decomposition into components.

Chapter 6 introduces a contract-based approach to assurance and structured expressions as a mean of formalize those contracts.

Chapter 7 presents how the designed case studies for three different domains were planned to evaluate the proposed methodology.

Chapter 8 details how the proposed approach has been validated by means of several experiments throughout case studies.

Chapter 9 summarizes the main contributions and publications of this work. In addition, this chapter provides some insights about further work.

Annex A shows further details about the analysis done on the standards

"Safety is 25% Common Sense, 80% Compliance and the rest is good luck" – Barry Spud

2

State Of The Art

2.1 Introduction

This work deals with the design and development of a common approach for compositional assurance on safety-critical systems.

As Fig. 3 shows, this work is placed joining three different research areas: Safety Standards Compliance, Assurance Cases and Components-based Approaches.

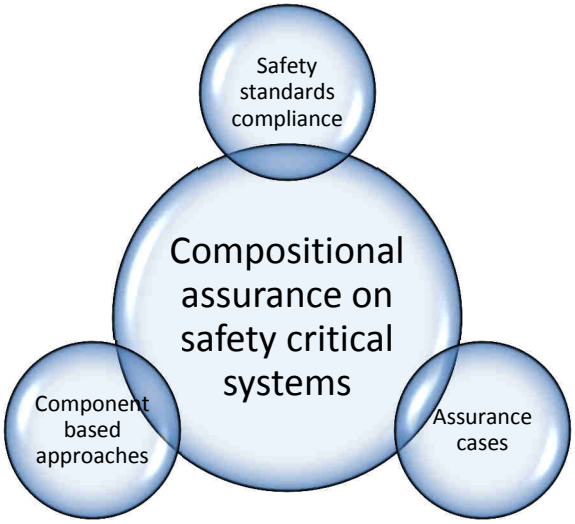


Fig. 3 Research areas related to this work

This work relies on different concepts from these areas. In order to clarify the context and the foundations in which this approach is based and to provide a basic background for understanding the overall thesis work, different concepts are introduced in this chapter.

Specifically, the rest of this chapter is organized as follows. Section 2.2 provides the main characteristics of the safety standards compliance for the safety-critical systems area. Section 2.3 provides an overview of the Assurance cases area and its principles. Section 2.4 presents the foundations of the component-based development for safety-critical systems.

2.2. Safety standards compliance for critical systems

Safety-critical systems can be defined as those whose failure could result in damaging either people or the environment. According to Knight [Knight 2002] “there are plenty of definitions of the term safety-critical system but the intuitive notion actually works quite well. The concern both intuitively and formally is with the consequences of failure. If the failure of a system could lead to consequences that are determined to be unacceptable, then the system is safety-critical”. In the previous chapter the actual situation of safety-critical systems and how industries deal with standard compliance on a regular basis have been introduced.

Knight speculates that one of the challenges safety-critical systems will have to deal is “comprehensive approaches to total system modelling be developed so that properties of entire systems can be analyzed. Such approaches must accommodate software properly and provide high fidelity models of critical software characteristics. They must also deal with the issue of assured non-interference”. Safety-critical systems need to ensure the functions are isolated and there is enough separation and independence between the involved elements so the functions do not interfere between themselves for an adequate performance.

Safety-critical systems are identified once an initial hazard and risk analysis has been performed, if safety concerns are discovered then it is requested to be certified [Storey 1996]. Rushby in [Rushby 2007] expressed that current certification practice is “standard-based” which implies that standards are responsible for prescribing the requirements and objectives the system should comply with. These “standard-based” certification is very costly and its objective it to provide stakeholders and society in general the confidence that the system will not suffer any unacceptable risk that could end in harm of any type. Certification requires that applicants shall follow prescribed processes and develop specific evidences.

On the Civil Aviation certification is defined as a legal recognition that a product, service, organization, or person complies with the requirements states in a certain standard. This implies technically checking the object of certification to very formally that complies with the applicable requirements. For certifying a product the authority should assess the design process of the product to ensure an acceptable level of safety, check whether the product actually conforms to the expected design and issuance a certificate required by the national laws to show the product has gone through the assessments process [DO-178c]. We can assume than when complying with a standard we are actually looking for a legal recognition.

Certification is applied to complete systems, while analysing down into subsystems. As Rushby mentioned in [Rushby 2007] the Federal Agency of Aviation (FAA), the USA authority responsible for certification avionics systems, only certifies complete aircrafts, engines and propellers. The FAA does not certify components such as operative systems, or air cruise control application outside the target airplane where they are used. It is only

considered the system as a whole and not the components or parts in which it can be decomposed. The analysis should be done considering the complete target system.

Safety-critical systems however, find it difficult to introduce new technologies, methods or tools in their design and developments as they have to go through the certification liaison process. Applicants should provide to the authorities the compilation of evidences from their result of their system safety assessment [Papadopoulos McDermid 1999]. While this might be easy and practical on those systems with long and extensive experience to the efficacy of the prescribed process and methods, this actually becomes a barrier for systems innovation where systems differ from previous ones or where new design, implementation or assurance techniques are applied [Rushby 2007].

Advisory circulars in the avionics domain are documents issued by the authorities like the FAA (Federal Avionics Agency) which works in United States or the EASA (European Aviation Safety Agency) in Europe, which are not standards, but are intended to provide guidance on accepted compliance means for specific challenging topics. For instance, the AC 20-148 [AC 20-148] provides recommendations concerning reusable software components. This advisory circular indicates that in order to reuse components, stakeholders must identify any installation, safety, operational, functional and performance possible concerns. Developers need to state clearly the DO- 178C [DO-178c] objectives that are fully and partially addressed, and how compliance has been achieved. They need to state clearly the failure conditions, safety features, protection mechanism, architecture limitations, software levels, interface specification and the process for certification. The AC 20-170 [AC 20-170] intends to lead the applicants into the idea of incremental certification. The incremental certification is linked to the incremental acceptance. This is the process for obtaining intermediate credit towards the final approval and certification. In each stage of the acceptance, the credit is obtained in a form of recognition so as to show for future use in a certification process. The mayor benefit of this incremental acceptance process is the ability to integrate and accept new components in the system without the need for re-acceptance of the previous integrated components. However, as stated before, certification is only issued at complete system level, instead that at component level.

Although in the automotive domain there is not a legal obligation to obtain the certification from the public authorities, the ISO 26262 functional safety standard is considered the state of the art[8]. In fact, the ISO 26262 standard for the automotive industry does not include the word “certification” in the whole document. One of the differences between the avionics and the automotive domain is that in the ISO 26262 standard the certification liaison process is not included. In Europe, the General Product Safety Directive (GPSD) 2001/95/EC applies in the absence of specific European regulations for safety of certain product categories and complements the provisions of

[8] The Draft International Standard (DIS) of ISO 26262 was published in June 2009. Since the publication of the draft, ISO 26262 has gained traction in the automotive industry. Because a public draft standard is available, lawyers treat ISO 26262 as the technical state of the art

sector legislation, which do not cover certain matters, for instance in relation to producers' obligations and the authorities' powers and tasks. As stated by lawyers [Klindt 2012] [Reuter 2012], ISO 26262 should be treated as published state of the art and so it should be complied in order to reduce the risk of liability.

In the automotive domain the concept of assuring the complete systems also applies. In ISO 26262, the functions at vehicle level are mapped to the "item" concept.

In ISO 26262 [ISO 26262] Development Interface Agreements (DIA) are described as a way to specify both procedures and responsibilities allocated to distributed developments for items and elements. The DIA includes information beyond technical safety by addressing procedural and confidence related issues. The use of DIAs is intended to help address risks such as: a supplier with inadequate capability, improper understanding or definition of the boundary of component and its interactions with its environment, or failing to fulfil requirements.

In ISO 26262, the term Development Interface Agreement (DIA) is used to define the procedures and responsibilities allocated within distributed developments for items and elements. In the DIA the supplier should exchange with the customer information such as: feedback about conflicts, completeness, consistency, etc.; technological limitations, behaviour models, incl. fault models, feedback about boundary between the Component and its environment. Those are the kind of properties that can be classified as in the scope of the safety contracts. However, the DIA includes more information that is not classifies as safety but also procedural and confidence related argumentation. The objective of the DIA is to intend to avoid as much as possible the risk from:

- a supplier with inadequate capability,
- improper understanding or definition of the boundary of Component C and its interactions with its environment,
- not fulfilment requirements of 5.4.4, as applied to hardware Component C.

In the medical domain, the manufacturer is the responsible of the final device which should go through the certification liaison process. Although the manufacturer may consider integrating developments done by suppliers, (s)he is still fully responsible for ensuring the product safety and the risk management activities have been executed and appropriate risk control measures have been applies event to the outsourced developments [ISO 14971].

In the medical domain reused is accepted by the use of SOUP components. In this case the manufacturer is expected to ensure:

- The product development methodologies used are appropriate. The FDA recooments to include an audit of the design and development methodologies used to thoroughly assess the development and qualification documentation generated .

- The procedures and results of the verification and validation activities performed to the component are appropriate and sufficient for the safety and effectiveness requirements of the medical device.
- There are appropriate mechanisms for assuring the continued maintenance and support of the components [OTS Guidance 1999].

One of the main differences between the avionics, automotive and the medical domain in relation with functional safety is how this concept is covered.

The automotive domain has integrated all functional safety concerns into one standard, ISO 26262. In this standard it is covered the product lifecycle from the concept level to the release, maintenance and operation of the product along with the management and supporting processes, as it is represented in Fig. 4.

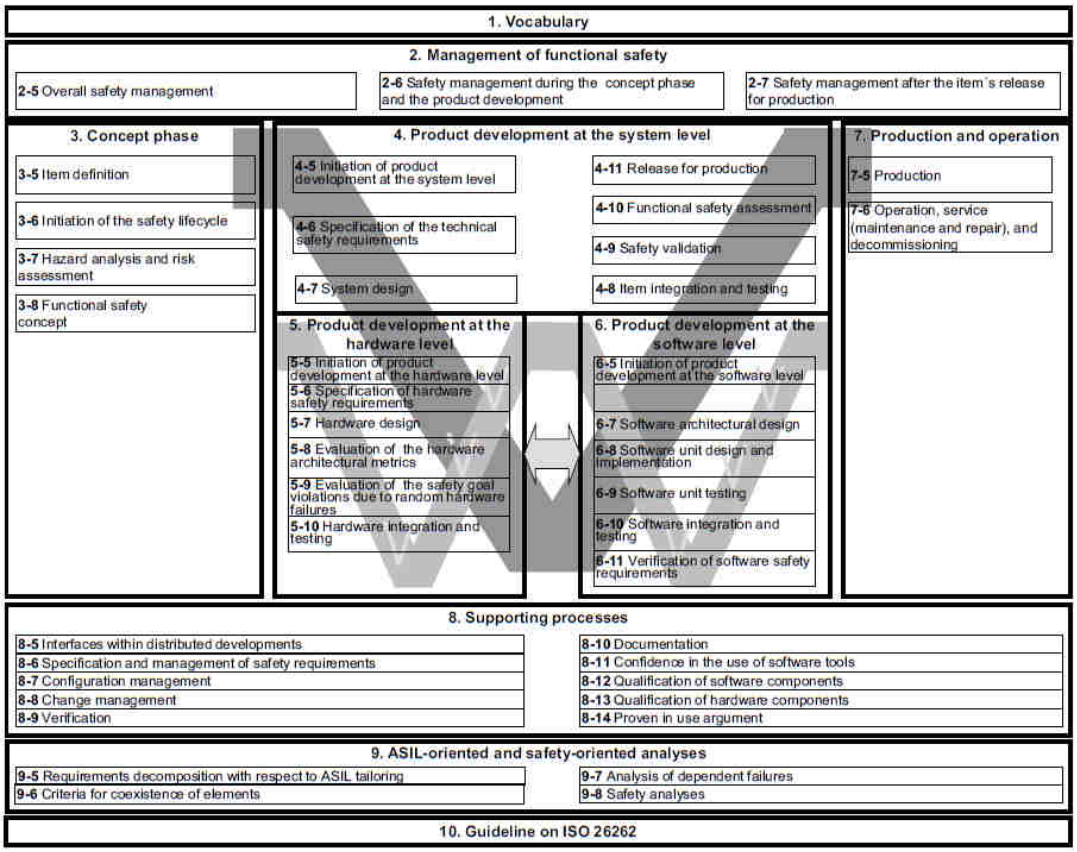


Fig. 4 ISO 26262 Overview [ISO 26262].

Avionics domain on the contrary has standardised the different phases of the development in different domain specific standards. In Fig. 5 the relations between the different avionics standards in the design and operational phase are identified.

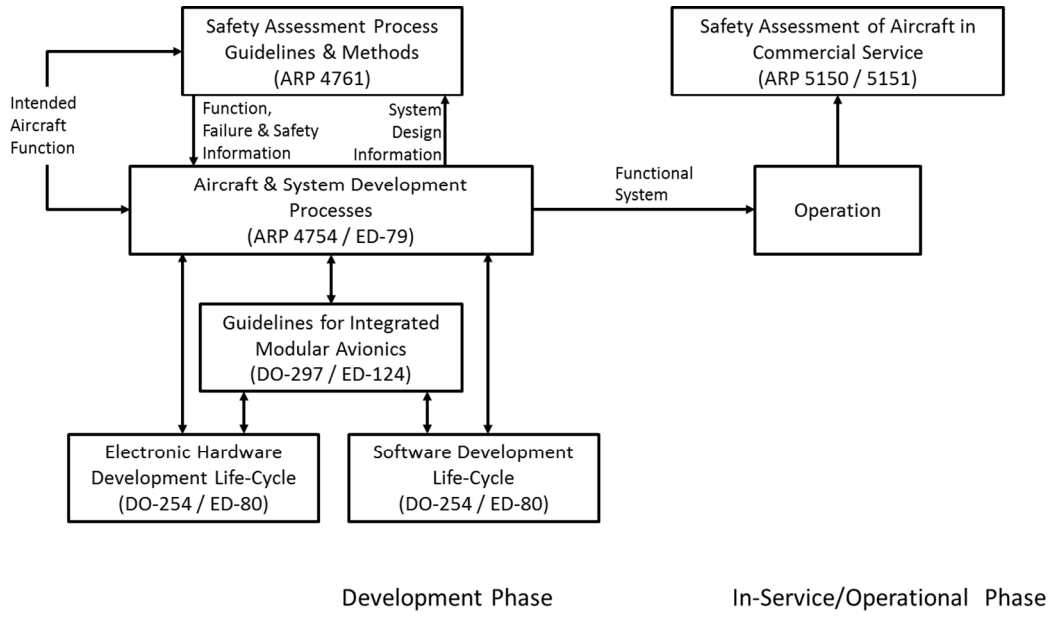


Fig. 5 Guidelines documents covering development and in-service/operational phases [ARP 4754A].

In [OPENCOSS D1.1] a good description is given for the avionics, automotive and railway certification framework.

[Zeller et al. 2014] analysed different safety standards regarding software safety assurance highlighting the following similarities:

- Common notion of safety and certification
- Linear progressing safety process with dedicated phases
- Combined hazard assessment and risk analysis to derive safety requirements
- Criticality levels as means to allocation safety (integrity) requirements to system elements
- Verification activities are driven by the safety requirements
- Safety case provides evidence that safety requirements are fulfilled which is needed for certification.

And they also identify the following divergences:

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

Regarding the medical industry, in Europe it is only requested for certification to be compliance with EC directives while the application of ISO standards are not mandatory just recommended, however, they provide strong bases while requesting certification and in some cases a mandatory requirement expressed by customers.

In Fig. 6 a collection of the standard related to functional safety for medical devices are described as described by Hobbs in [Hobbs 2011].

In the avionics domain when a critical hazard occurs, the safe state will be to land as soon as possible and finish with the aircraft operation. In a road vehicle there is no need for landing so the system tends to shut down when a hazard occurs. However, in medical regulation, safety is defined as "the probable benefits to health for its intended use when accompanied by adequate directions and warnings against unsafe use, outweigh any probable risks" [OTS Guidance 1999]. In this sense the safe state has a different concept behind and also the way in which the requirements are derived. The safety requirements derived for the safe state are not related to the normal behaviour and expected functionality but to the abnormal behaviour and performance in presence of a fault.

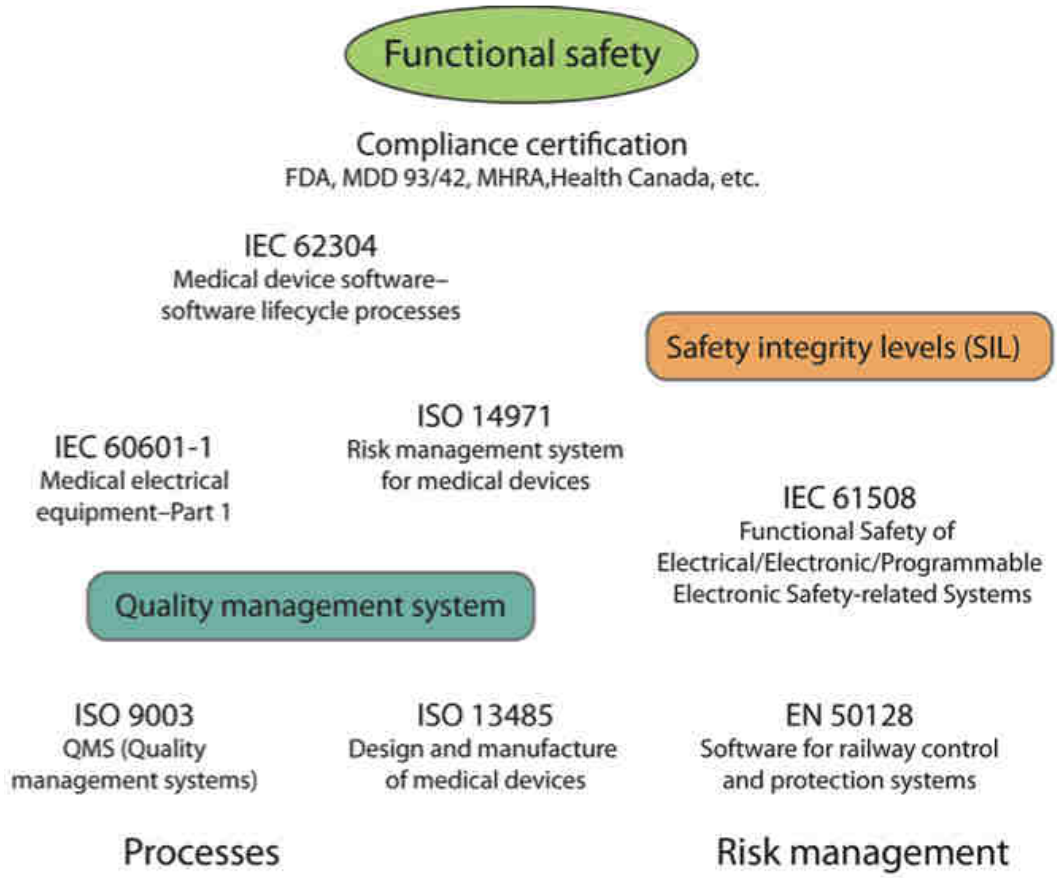


Fig. 6 Some of the standards contributing to functional safety in medical devices [Hobbs 2011].

[Dodd Habli 2012] describes the audits known as Stage of Involvement (SOI) done at strategic point in the software lifecycle with the aim of reducing the risk of failing the final certification audit. The earliest a potential certification failure is identified the better. A failure will normally consists of reworking an artefact requested by the authority before the audit is repeated and directly affecting the final cost. That is the reason SOIs are planned frequently with a typical time duration between audits is four to six months. In fact, the idea of periodic SOI is similar to the idea of early validation and verification in order to reduce the possible project deviations.

Dodd and Habli distinguish two types of certification: prescriptive certification and goal-based certification. In the first one, applicants show that system is acceptably safe based on 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. Avionics certification follows the prescriptive certification approach while the automotive and medical safety standards fit better on the goal-based approach.

[Kelly et al. 2005] analysed the goal-based certification approaches where challenges are identified as follows. It is hard to find skilled and experienced people to enable the potential benefits of the goal-based standards. Kelly says that in the defence context it is possible to find this technically qualified people but it is necessary not only in this context but also a similar level of qualification in the customer community. It is a challenge not only in finding but also in preserving the skills and ensuring consistent decision making in long running projects.

Prescriptive-based certification is not free from criticisms, [McDermid 2001] [Redmill 2000] the use of qualified tools, techniques and methods as the ones mentioned in the standards do not necessary mean that the system will achieve the desired level of integrity.

Standards-based approaches are defined as those which follow the prescribed process and released prescribed documentation. “These approaches work well in fields that are stable or change slowly” [Rushby 2010-1]. In order to cope with this, Argument-Based Approach to Certification is proposed. This is the one where applicant develops a safety case, whose outline form may be specified by standards or regulation. It makes an explicit set of goals or claims and provides supporting evidence for the claims and arguments that link the evidence to the claims.

With the idea of inheriting the better of the two approaches, a hybrid approach is presented in [Stensrud el at 2011] by integrating prescriptive elements from the standards into a goal-based safety case. Their approach is based on transformation, from tables presented on the standards into claims. Prescriptive standards requested that certain analysis, methods, techniques or activities to be perform during the development. These analysis, methods or activities are recommended or highly recommended based on the critical level for the function and grouped on 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.

There have been different attempts to model the standards. In eDIANA project (Embedded Systems for Energy Efficient Buildings) [Arana et al. 2011] a Software Engineering Process for Certification Metamodel is proposed. They created a general certification “language” by means of a structured semi-formal meta-model, which acts like a template for certification requirements specification. Fig. 7 describes the inputs for the meta-model creation and how it was used.

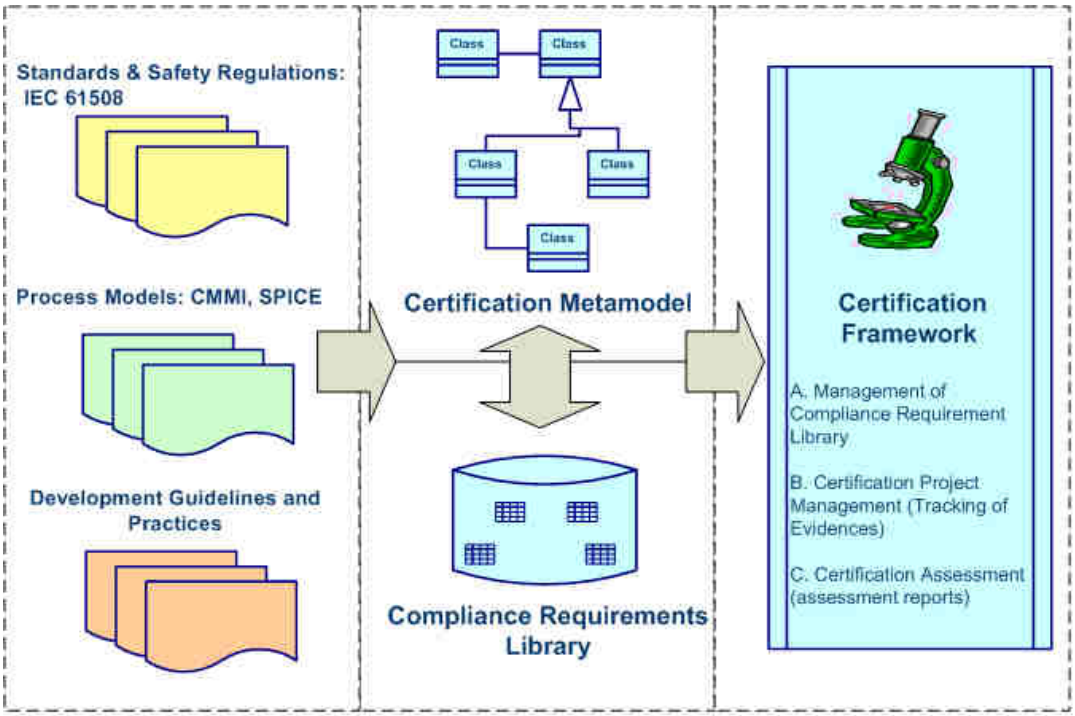


Fig. 7 Derivation of the Certification Meta-model by Arana [Arana et al. 2011].

Another project, ModelMe! (Model-Driven Software Engineering for the Maritime and Energy Sectors) also proposed another approach. Some of their conclusions were in three main aspects: Safety Evidence Management, Safety Assessment (Modus) and Traceability and Slicing.

- The Safety Evidence Management aspect described in [Panesar-Walawege et al. 2010] focused in improving system assessment and development practices through the application of Model-Driven Engineering. In Fig. 8 we can see and except of the IEC 61508 standard meta-model. The complete model includes 124 concepts, and 32 association types. Specifying safety standards requirements through modelling seems worthwhile and brings many practical benefits.
- They developed Safety Assessment methodology called Modus [Sabetzadeh et al. 2011] aims for quantitative assessment of dependability cases. Many safety decisions are based on expert judgment and assurance cases[9] could be a solution for showing these decisions. The concept of assurance case will be developed further in section 2.3. Assurance cases
- Regarding the Traceability and Slicing aspect, requirements-Design traceability is specially challenging as is a major observed issue in certification. For that reason improving traceability and using traceability for automated analysis (slicing,

[9] Assurance Case is a set of auditable claims, arguments, and evidence created to support the claim that a defined system/service will satisfy the particular requirements [SACM 1.1].

impact analysis, etc) are some important concerns for suppliers and certifiers alike. In [Falessi et al. 2011] they proposed the use of a tool SafeSlice for creating traceability links and checking consistency. Their approach is based on SysML system models where by Slicing of design models they are able to manage safety inspections and report generation. This is important in order to check components properties.

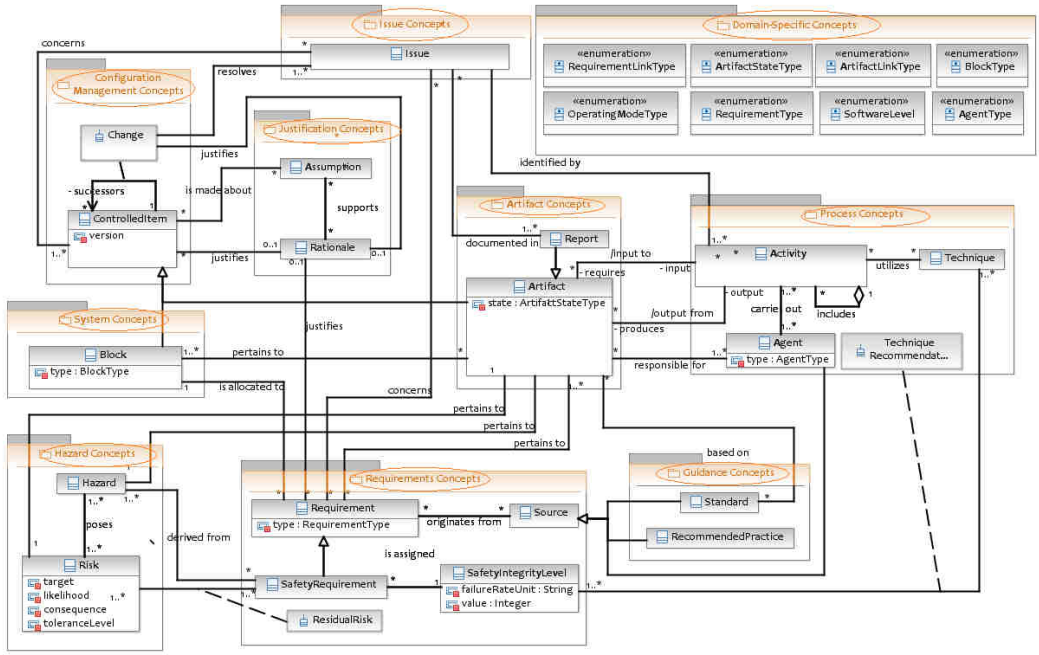


Fig. 8 Excerpt of the Conceptual Model of IEC 61508 resulting from ModelMe! project

In the automotive industry there has been an interest on developing a domain specific language called EAST- ADL [EAST-ADL V2.1.12] which purpose is to model with enough detail the automotive electrical and electronic systems so as to be able to generate documentation, design, analysis, and synthesis. For doing so it is requested to have system description at different layers of abstraction and detail. These activities also involve specifying non-structural aspects of the electrical/electronic system under development, such as requirements, behaviour, and verification and validation.

In EAST-ADL they include the dependability package in order to provide support for safety information organization according to ISO 26262. The Dependability package (see Fig. 9) includes support for defining and classifying safety requirements through preliminary Hazard Analysis Risk Assessment, tracing and categorizing safety requirements according to their role in the safety life-cycle, formalizing safety requirements using safety constraints, formalizing and assessing fault propagation through error models, and organizing evidence of safety in a Safety Case.

This package tries to provide some support for ISO 26262 requirements, specifically for part 3.

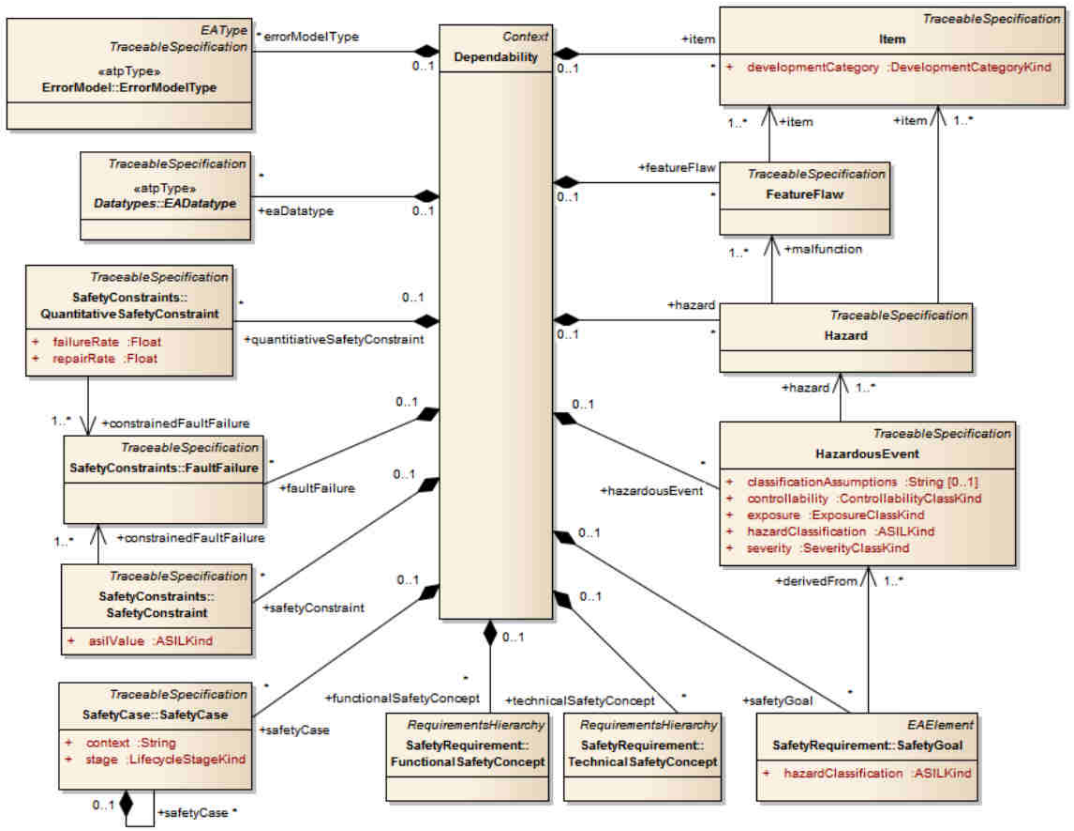


Fig. 9 Diagram for organization of dependability related information [EAST-ADL V2.1.12].

2.3. Assurance cases

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" [Def Stan 00-56]. However, these cases are not just limited to the safety area; we can find the previous concept applicable to IT trustworthiness [Gorsky 2004], security [Bloomfield et al. 2006] 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" [SACM 1.1].

Wagner [Wagner et al. 2010] structured safety cases into three parts "(1) the safety goal that has to be achieved, (2) the available evidence for achieving this goal, and (3) the structured argument, which establishes the systematic relationship between the evidence and the goals"

Bloomfiel [Bloomfield 2012] talks about three argumentation approaches:

1. The goal-based approach where the safety properties which are satisfied are argued.

2. The rule-based approach where the argumentation is focused on standards compliance and
3. The risk informed approach where the argumentation is centred in vulnerabilities and hazards mitigated.

Flood and Habli [Flood Habli 2011] also make an argument categorization defining:

- risk (or "primary") arguments – that aim to establish that the system is acceptably safe to be deployed.
- confidence (or "backing") arguments – that are used to justify that sufficient confidence can be placed in evidence and inferences of the risk arguments
- compliance arguments – that show that requirements of the applicable standards have been satisfied.

In relation with the previous section where we described the safety standards compliance the work of Holloway [Holloway 2013] is relevant as he creates the assurance case implicit for the DO-178C. He categorizes the objectives into: (i) The objective is likely to appear in some form as a claim or evidence in the primary argument. (ii) The objective is likely to appear in some form as a claim or evidence in a confidence argument.(iii) The objective is likely to appear as context, assumption, or justification in an argument rather than as a claim or evidence.

However, safety cases are seen with criticism due to several reasons [Johnson Robins 2011], [Johnson Derek 2011]. One of the main complaints against safety cases is that it is always possible to find or produce evidence that something is safe. It is the confidence level that is put into that evidence what gives strength to the argumentation. Unfortunately, for safety analysis there is no complete mathematical theory to base arguments and guarantee completeness. Bloomfield [Bloomfield 2012] and Rasche [Rasche 2001] also highlighted following main issues while applying safety cases:

1. The tendency for practitioners is to use extended expansions for claims that cover many issues which are often very hard to justify even informally.
2. Architecting assurance cases is a specialised activity and some of the current assurance cases should be review it and will need significant rewriting to apply an architecture structure.
3. There is a large gap between the practice and the best practices; there is a lack of guidance on claims-argument-evidences.
4. The amount of work required to construct a safety case including the specialized and costly (outside) resources required.
5. Problems associated with obtaining and validating data to justify a probabilistic risk analysis.
6. Too much focus on technical risk and not enough on meeting the needs of workers.

The Nimrod report [Haddon-Cave 2009] also includes some criticisms to safety cases indicating that just by doing a safety case, it will not imply that the safety case is correct and too complicated safety cases will induce to produce incorrect safety assessment.

Unfortunately, there is not a method from preventing in given inappropriate argumentation. The ideal scenario for creating strong, complete safety cases is to provide an independent, non-subjective argumentation. This could be reached by demonstrating that major hazards of installation and the risks to personnel therein have been identified and appropriate controls provided.

It is not rare, while doing safety assessment, to be presented long reports referencing to evidence, but those reports lack in clarity on how that evidence relate to the safety requirements and how it is understood to comply with the standard. In fact one of the complaints Haddon Cave made on the Nimrod report is that "Safety Cases and Reports are too long, bureaucratic, repetitive and comprise impenetrable detail and documentation".

When creating a safety case, the argumentation defined as a connected series of claims intended to establish an overall claim is required to be shown. In attempting to persuade others of the truth of an overall claim, we make supporting claims which also need to be supported; giving us a hierarchy of claims. Ultimately these claims should be supported by evidence.

In Fig. 10 we show how the top claim is decomposed into two sub claims (claim 1 and claim 2). Claims should be stated as true/false evaluated proposition.

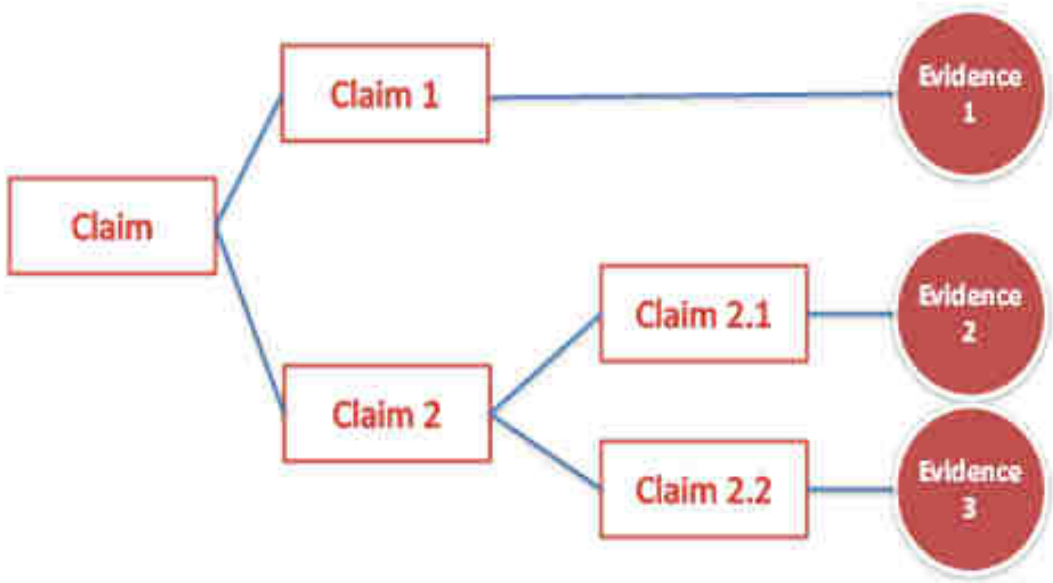


Fig. 10 Argument decomposition logic

To inference that proposition stated in Claim is truth the following logic should be applied:

```
IF Evidence 1 THEN Claim 1
IF Evidence 2 THEN Claim 2.1; IF Evidence 3 then Claim 2.2
IF Claim 2.1 AND Claim 2.2 THEN Claim 2
IF Claim 1 AND Claim 2 THEN Claim
```

Kelly [Kelly 1998] proposes to use argumentation to justify the rationale behind the design decision that a system is safe for a specific environment and under certain circumstances. He defines six steps in the top-down development of a safety case:

1. Identify the goals to be supported;
2. Define the basis on which the goals are stated;
3. Identify the strategy used to support the goals;
4. Define the basis on which the strategy is stated;
5. Elaborate the strategy (and proceed to identify new goals–back to step 1), or step 6;
6. Identify the basic solution.

This methodology has been accepted by the GSN community [GSN Standard] and it has been even adapted for developing from the bottom up:

1. Identify evidence to present
2. Infer "evidence assertion" claims to be directly supported by these pieces of evidence, and present these as claims;
3. Derive higher-level sub-goals that are supported by the evidence assertions;
4. Describe how each layer of sub-goals satisfies the parent goal (i.e. strategy);
5. Check that any necessary contextual information is included;
6. Check back down the structure for completeness;
7. Join the resulting goal structure to a known top goal or a set of sub-goals.

Wagner [Wagner et al. 2010] proposes to structure the product-related safety case into arguments about the system itself in different abstraction levels and arguments about the environment and the user of the system. He states that by identifying generic safety case modules and several reoccurring patterns, this can be reused supporting the development of future automotive safety cases.

Similarly, [Papadopoulos McDermid 1999] also worked on specifying a process for safety assessment but their particularity is that they proposed a common process for system development and assessment which could be acceptable in the framework of each safety standard in consideration. Their model consists of three integrated processes: a development process, a safety assessment process and a safety case process.

Denney and Pai even go further and proposed a lightweight methodology [Denney Pai 2012] to give systems engineers a capability to (i) continue to maintain the existing set of artefacts, as per current practice, (ii) automatically generate (fragments of) a safety case, to the extent possible, rather than creating and maintaining an additional artefact from scratch, and (iii) provide different views on the relations between the requirements and the safety case. They implement a tool to support this methodology and where they can assist in a semi-automated safety case generation where they make the following transformation from the design model to the safety case:

- hazard, requirement, causes → goal, sub-goal
- allocated requirements →sub-goals

- mitigation, verification method → strategy
- verification allocation → evidence
- requirement source, allocated artefact → goal context

In relation to this semi-automatic generation of the safety case, another important idea is introduced in [Stensrud et al. 2011]. They presented a hybrid approach between prescriptive standards and safety cases. The idea is the appliance of safety argument patterns by transforming a SIL (Safety Integrity Level) table into a GSN safety pattern. SIL tables appear on the IEC 61508 standard in order to show the mechanism, activities applicable depending on the criticality of the function. Stensrud proposes a different approach to safety case patterns, where patterns are related to certification objectives and how they help introducing conformance items for the IEC 61508 standard on safety cases and improving transparency in certification processes. Their main interest is to take advantage of the goal-based structure of the safety cases and integrate the prescriptive elements of the standard so as improve the transparency and consistency of the safety certification.

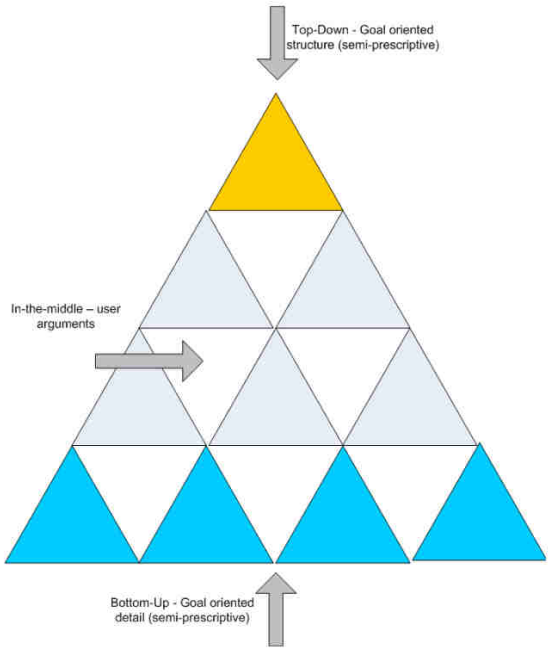


Fig. 11 Schematic illustration of a safety case [Stensrud et al. 2011].

Stensrud work is based on the presented a safety case framework defined by Weaver [Weaver et al. 2002] that includes the top level software safety argument where the top level goal is that the system is acceptably safe. The top level goal is further broken down into sub goals including that the safety requirements are valid. Fig. 11 describes this safety case framework. Stensrud address the middle level of the safety case (the grey triangles in the figure). They propose a collection of safety case patterns derived from the IEC 61508. However, the link between their patterns with the upper pyramid patterns and still is just being used on one standard, the IEC 61508.

Kelly and McDermid [Kelly McDermid 1997] introduced the idea of safety case patterns a mechanism used to reuse some parts of the safety case that support its construction. They can be used to apply the best practices of argumentation. Concerning this [Sutcliffe Carroll 1999] mentions "Claims as a means of reuse will have to face the problems encountered in software component reuse, such as obtaining a critical mass of claims to persuade designers to buy into the process of design by reuse, the not-invented-here syndrome, need for management incentives and legal and copyright issues".

There are some argumentation patterns catalogues published in [Stensrud et al. 2011] [Matsuno et al. 2010].

In order to achieve the challenge of complexity and length of the safety cases, the idea of modular safety cases appears. By adopting a modular, compositional, approach to safety case construction it may be possible to:

- Justifiably limit the extend of safety case modification and revalidation required following anticipates system changes
- Support (and justify) extensions and modifications to a 'baseline' safety case
- Establish a family of safety case variants to justify the safety of a system in different configurations.

This approach establishes a modular and compositional construction for safety cases that has a correspondence with modular structure of the underlying architecture. As with system architecture it would need to be possible to establish interfaces between the modular elements of the safety justification such that safety case elements may be safely composed, removed and replaced. Similarly, it will be necessary to establish the safety argument infrastructure required in order to support modular reasoning.

Kelly in [Kelly 2001] affirms that "By adopting a modular, compositional, approach to safety case construction it may be possible to:

- Justifiably limit the extend of safety case modification and revalidation required following anticipates system changes
- Support (and justify) extensions and modifications to a 'baseline' safety case
- Establish a family of safety case variants to justify the safety of a system in different configurations."

Once a modular safety case is created, a module can be reference in other parts of the safety case or be reused along with the system associated.

Despotou highlighted the following advantages of using modular safety cases approach [Despotou Kelly 2008]:

- Reuse of arguments
- Containment of impact of change
- Contracts between argument modules provide additional barrier to propagation of change if the public (cross-referenced) goals from a supporting argument have changed.

- Limiting the cost of (re)generating evidence
- Integration of process and product arguments
- Standardisation of processes

He also addressed two challenges: (1) Complexity, in relation that reference between modules could increase complexity and reduce clarity. In this sense, coupling between arguments should be maintained at reasonable levels. (2) Loss of uniformity, and over-specification, in relation to the level of abstraction of the modules. Argument contracts should make clear the relationship of the referenced arguments.

The GSN Standard has created an extension in order to cope with the new concepts that modular safety cases have to deal with [GSN Standard] such as:

- Away Goal: This repeat a claim presented in another module which is used to support the argument in the local module.
- Module reference: Presents a reference to a module containing an argument
- Contract module reference: Presents a reference to a contract module containing definition of the relationships between two modules, defining how a claim in one supports the argument in the other.
- Away Solution: It repeats a reference to evidence items presented in another argument module.
- Away Context: It repeats a contextual artefact.

[Palin Habli 2010] proposes the use of modular safety cases on its use for automotive domain. Palin's approach starts by specifying top level safety claims for automotive systems, proposes argument strategies and evidence that can substantiate the safety claims (using ISO 26262 as context, where appropriate); and finally, he proposes an argumentation framework based on the reuse of argumentation patterns. Palin created a pattern catalogue of automotive safety arguments (Fig. 12) The argument patterns are identified, some of which are designed to be connected together to produce integrated product and process arguments. The patterns address aspects of safety related to safety requirements, hazard/risk analysis and through-life safety

Another challenge identified in [Sutcliffe Carroll 1999] is the misinterpretation due to the use of natural language: "claims have an advantage in being a structured natural language description which facilitates communication between users and designers from different communities. The penalty in natural language is the potential for misinterpretation". They also propose some recommendations to support the reuse: claims should be indexed and formatted so retrieval mechanism can be used when reusing in new design context. The use of faceted classification schemes for the indexation and retrieval has already been applied for indexing reusable software components, but with limited effectiveness. Sutcliffe suggests that task-artefact cycle as a more powerful approach. In this approach claims are associated to context scenarios where they have been used. These scenarios are as well associated with identified cases and classes of users, usage tasks and product

features. Sutcliffe affirms that this approach provides a richer classification scheme for interpreting how claims may be reused.

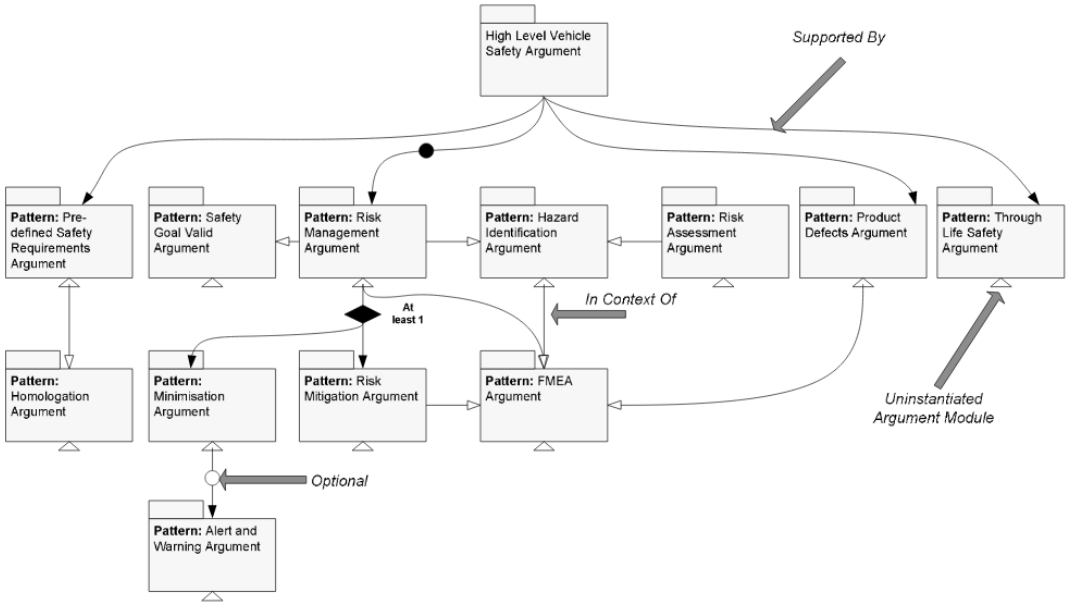


Fig. 12 Architecture for the Argument Pattern [Palin Habli 2010].

Rushby [Rushby 2010-2] also proposes to formalise the content of the claims in safety cases: "formalization of some elements may allow the context for human reviewers (e.g., assumptions) to be more precisely articulated and checked." In order to deal with the possibility of ambiguity arguments or arguments that tends to possible misinterpretation, he indicates that safety argumentation can be logical deduction, probabilistic, expert judgement or historical experience. Just by formalizing some elements of the safety case it will support precision and checking methods could be applied. He proposes to formalise of the content of the claims in safety cases.

Holloway [Holloway 2013] worked on the creation of the assurance case implicit on the DO-178c. He categorized the objectives into:

- The objective is likely to appear in some form as a claim or evidence in the primary argument.
- The objective is likely to appear in some form as a claim or evidence in a confidence argument.
- The objective is likely to appear as context, assumption, or justification in an argument (rather than as a claim or evidence).

There are coordinated efforts are currently underway in the International Standards Organization (ISO) and the Object Management Group (OMG). At the OMG we can identify the System Assurance Task Force (SysA) which goals are:

- Facilitate the development of a specification for a Software Assurance Framework

- Enable industry to improve visibility into the current status of software assurance during development of its software
- Enable industry to develop automated tools that support the common framework

The SysA group is behind the development of OMG SACM standard (Structured Assurance Case Metamodel) [SACM 1.1] which combines previous OMG specifications

- ARM (Argument Metamodel)
- SAEM (Software Assurance Evidence Metamodel)

The object management group (OMG) is working on standardizing the Structured Assurance Case Metamodel [SACM 1.1]. Standardization will ensure that end users are investing not just in individual tools but also rather into a coordinated strategy.

The objective of SACM is to provide a modelling framework which will allow users to exchange their argument structure. This is important from the tooling perspective as tools could interoperate so as to exchange information on the same concepts and be understandable by all. It reduces ambiguity. However, the representation of an argument in SACM does not imply that the argument is complete, valid, or correct. Similarly, the evaluation or acceptance of an argument by a separate party is not covered by the SACM.

An argument is usually defined as a series of linked premises (propositions), leading to a conclusion. From this we can derive a set of practical modelling approaches that allow users to link propositions (claims) together and to communicate how they consider that higher level claims are supported or derived from the lower level claims. In the SACM model, structured arguments are derived from that principle where argument elements (primarily claims) that are being asserted by the author of the argument, together with relationships that are asserted to hold between those nodes.

Another attempt related to the OMG is the Machine-checkable Assurance Case Language proposal, to get assurance cases that are machine checkable which implies to have a grammar for the assurance cases and reduce the ambiguity. However, this proposal is still under revision [MACL 2013].

2.4. Component-based development for critical systems

The first concept to be explained here is the composability. Maybe one of the most adequate definitions for this that applies to our context is the one done by Rushby "Composability means that properties of subsystems are preserved under composition" [Rushby 2007].

Component-Based Development (CBD) is becoming an increasing trend on complex system development. Some of the benefits associated with this practice are: reusability, maintainability, accuracy, clarity, replaceable, interoperability, scalability, performance, flexibility, adaptability, and reliability [Woodman et al. 2001].

The appliance of CBD also implies new capabilities that developers should fulfil. According to [Brown 1998] developers of large-scale and mission-oriented applications require many additional capabilities including:

- Re-engineer legacy applications to harvest existing components reusable in other applications or replaceable by newer technologies.
- Find suitable components both locally and externally.
- Integrate components implemented in a variety of different technologies.
- Validate a component's behaviour before using it.
- Manage multiple implementations of the same component in different technologies, and as it evolves over time.

When the CBD is applied to safety-critical systems then issues arise. As it is mentioned in [Rushby 2010-1] components are certified only as part of an airplane or engine. That is because the interactions is what matters and it is not known how to certify these composabilities. At the same time modern engineering and business practices use massive subcontracting and component-based development that provide little visibility into subsystem design.

When analysing accidents involving critical systems such as the Mars Polar accident [Mars Polar Loss 2000], we discovered that it is not a problem for component failure but for component interactions. Component failure accidents have received the most attention in engineering, but component interaction accidents are becoming more common as the complexity of system designs increases. Levenson [Leveson 2012] defines a component interaction accident as those ones which arise due to interactions among systems components (electromechanical, digital, human, and social) rather than in the failure of individual components. It is important to provide enough information about a component that it is possible to perform an analysis on component interactions.

When applying CBD to system engineering the following lesson learned should be considered:

- It is important to document the dependencies and effects making assumptions made by component designers and developers explicit [Sutcliffe Carroll 1999].
- It is recommended to have tool support for the Certification Process, especially when reusing software components [Illarramendi et al. 2014].

Regarding safety-critical systems, lately different projects have proposed different structures in order to support the component base development.

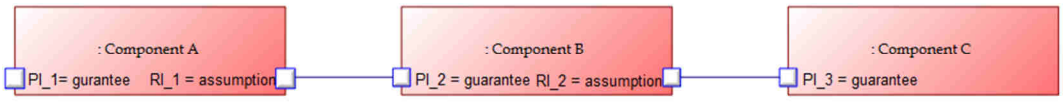


Fig. 13 Match between design-contracts proposed by ASSERT project

The European ASSERT project proposed to model the assumptions of a component as provided interfaces and required services as guarantees [Cancila et al. 2010.]. Components will interact only when their respective contracts are valid and the transitive closure over assumption is met. Fig. 13 illustrates this interaction. If we just consider Component A and component B , Component A guarantees some services (PI_1) with

attributes declared in the contract if its assumption, RI_1, is satisfied. This required interface RI_1 is satisfied by the guarantee PI_2 provided by Component B, (assuming the two contracts locally match).

However, we still need Component C to ensure that the overall system guarantees PI_1. This is because PI_2 requires RI_2 to be satisfied. We now have a chain from PI_1 to PI_3 where the last element (PI_3) is not dependent on any further required interfaces. Consequently, the system guarantees P1_1.

The CHESS project evolves from the ASSERT concepts. [Vardanega 2009] extending the application from the initial spatial domain proposed in ASSERT to aero-spatial, telecommunication and railway application domains strengthening results on both contract-based approach and mathematical structure.

SPEEDS was a European project which main objectives were:

- Modular (component-based) system design with multi-domains in a distributed partner environment
- Integration of domain/tool-specific design models into one tool-independent "Complete Virtual System Model"
- System analysis and simulation across domains, based on a "Complete Virtual System Model"

SPEEDS developed and implemented a formal meta-modelling language and the syntax of component. This language provides the format for a Complete Virtual System Model that integrates all domain-specific and tool-specific system models. As a result of the project not only the complete system model comes out but also the component's contract. These contracts identify the premises and promises of the component in order to behave in a specific way and an attribute designating its view-point. A viewpoint has no formal semantics but is used as a means of sorting contracts across a complete system specification. The specification of the assumption and promise assertions is actually the core of the contract; it presents a required capability of the component (associated with the viewpoint) [SPEEDS D.2.5.4]. These contracts define the premises and promises of the component regarding its behaviour in a specific way as well as an attribute designating its viewpoint.

CESAR [CESAR D_SP1_R3.3_a_M3] was another European project which came out with a meta-model (CMM: Cesar Meta-Model) as a result that offers a domain and tooling independent modelling capacity which can be used as extension points for domain and tool-specific extensions. The CMM includes the concept for Rich components, which can be connected and integrated on hierarchies. There can be different kinds of rich components such as operational actors, functions, logical components or technical components depending on the respective perspective. The CMM is based on an integration of component-based design with contracts based on input from SPEEDS project, EAST-ADL2 (traceability, verification and validation) from ATESST project and the

own CESAR Requirements Management Meta-Model (RMM). All these propose a metamodeling language for components dealing with safety-critical systems.

Another relevant work is the VerSaI concept introduced by Zimmer [Zimmer 2014]. Zimmer introduced a model based language used for contracts with its focus on vertical and horizontal interferences. The concept of Vertical and Horizontal Interfaces was first introduced by [Zimmer et al. 2010]. The authors distinguish between two types of interfaces; the Vertical Interface, which happens between the application and the underlying platform, and the Horizontal Interface that takes place between applications (whether they run in the same platform or not).

It is important that components do not include only functional specification but also include the behaviour when a malfunction appears. Related to this idea and the modular safety cases mentioned on previous section, the idea of safety case contract appears.

The IAWG (Industrial Avionics Working Group) consortium has been researching how to evaluate safety on a modular approach. Modular and incremental certification is seen as a strategy to deal with the cost of re-certification of change in relation with size and complexity of the system. According to Fenn [Fenn et al. 2007-1] there are two concepts we need to deal on modular software safety contracts:

- Dependency – Guarantee Relationships (DGRs): They capture guaranteed properties of a software component and define the properties on which that component is dependent in order to uphold its guarantee
- Dependency – Guarantee Contracts (DGC): The relationship that captures the dependencies from one software element that may be satisfied by the guarantees provided by other element.

A safety case contract is seen as the mechanism to record the interdependencies existing between the argumentation modules that form the safety case. These contracts are used to shown how the claims from one module are supported by arguments from another module. Kelly in [Kelly 2001] proposed a tabular form in order to match successful link between two or more modules of the safety cases. However, the Industrial Avionics Working Group (IAWG) found some problems:

- It was unclear without more explicit examples, what exactly the safety case contract table was meant to cover, and how it was to be applied. In practice it was found to be difficult to capture all the necessary information in such a tabular form.
- There is no mechanism for capturing the strategy used in addressing one goal with another. This strategy could in many cases be fairly complex. In the same way that strategy (potentially with its own context and assumptions) may be needed to show how a goal within a module solves another, this may also be required where the solution is made across modules via the contract.
- The tables exist as completely separate entities from the GSN argument itself. This means that here is no visibility with in the GSN structure of contractual links

Fenn [Fenn et al. 2007-2] proposed to take advantage of the GSN graphical notation and safety cases argumentation within the safety case contract as it provides more expressiveness and clarity than the tabular approach and also be integrated with the safety case argument. The GSN argumentation notation enables to capture the rationale behind the safety contracts relationship, where an "away goal" requiring support in one module, cannot be directly mapped to a "public goal" elsewhere. This way strategies, justifications, and context are also included on the contract and the rationale is made explicit. Fenn also proposes a generic pattern for safety case contract modules (See Fig. 14). They indicate that dealing with context can be complex in the practices as contexts inherited from components that have been developed independently are unlikely to be equal and the differences are even greater if they have been developed in different domains. They also proposed an argument pattern to be used for the safety contracts that was used on their case studies.

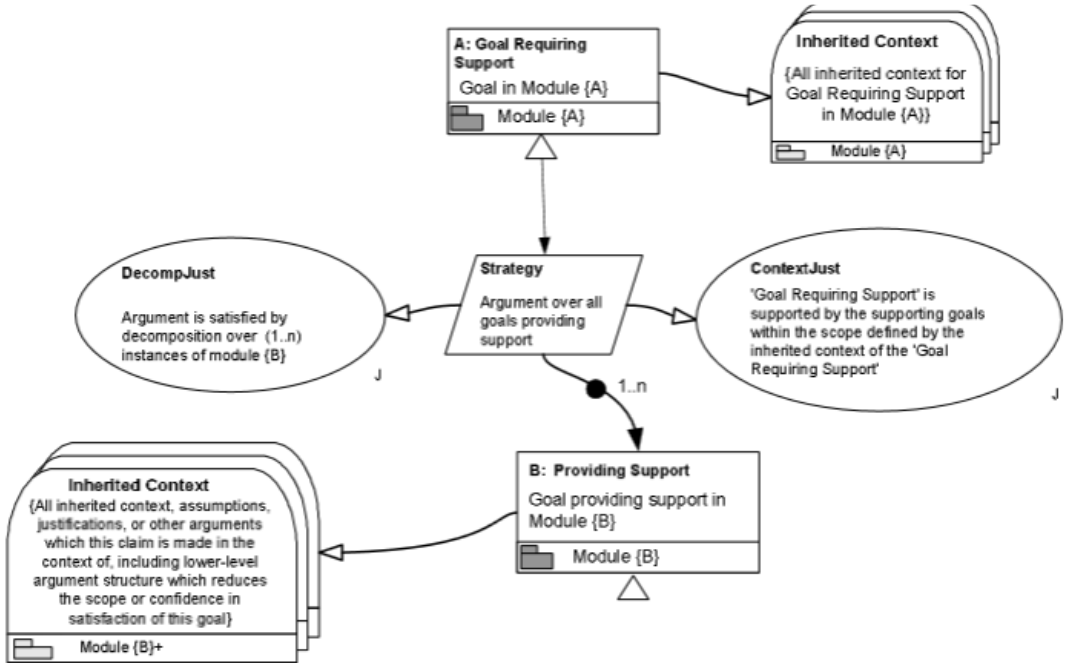


Fig. 14 Generic pattern for safety case contract modules [Fenn et al. 2007-2]

Safety case contract-based approaches have been proposed for certification of COTS-based systems [Ye Kelly 2004]. Ye and Kelly propose the use of a contract that "must record an account of the match achieved between the objectives required by the application argument module and addressed by the COTS component argument module. In addition the contract must also record the collective context agreed as consistent between the participant modules".

Conmy proposed a method [Conmy et al. 2003] that can be seen in Fig. 15 an Integrated Modular Avionics system which is a component-based architecture used in avionics. The process is based on the analysis of each component in the context of the overall system design and then finding derived safety requirements. Each IMA component (hardware,

software or both) is then examined to determine how these safety requirements are met, and a contract is formed which captures the rely/guarantee conditions between that component and any component which relies on it.

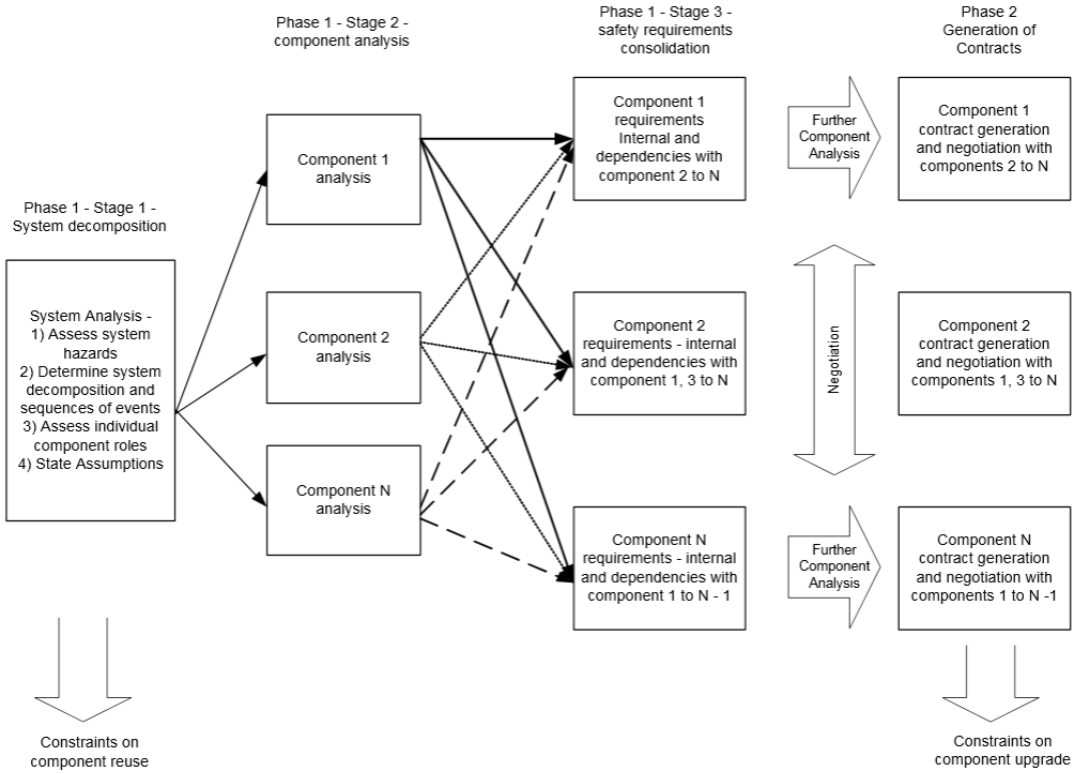


Fig. 15 Summary of the safety process proposed in [Conmy et al. 2003]

From another perspective, the FRESCOR project (Framework for Real-time Embedded Systems based on COntRACTS) [FRESCOR] proposed contract-based resource management in distributed systems. It uses service contracts as a mechanism for dynamically specifying execution requirements. To accept a set of contracts, the system has to check as part of the negotiation whether it has enough resources to guarantee all the minimum requirements specified, while upholding guarantees on all previously accepted contracts negotiated by other application components. If successful, the system reserves enough capacity to guarantee the requested resources and will adapt any spare capacity available to share it among the different contracts that have specified their desire or ability to use additional capacity.

The SAFECER project also made some work in relation to composition. They proposed the use of composition contract, “a composition contract can be formulated to establish relations between their respective contracts and properties” [SAFECER D2.2.1]. In this context Sljivo [Sljivo et al. 2013] introduced the concept of weak/strong assumptions/guarantees when formalizing for addressing a broader component context and specification of properties for specific alternative contexts. They propose contracts in which all properties that an environment shall satisfy are defined separately from those

required only in some contexts. This allows their contracts to be used for components and in different contexts. Their proposal includes the specification for all the properties that an environment must satisfy separately from the guarantees and assumptions that are required to hold only in some contexts. The latter is specified as a set of weak A=G pairs to preserve the connection between assumptions and guarantees, and to enable specification of additional properties for specific alternative contexts.

When a component is integrated in a new context the formal specification proposed by Sljivo can be used to check consistency of component integration contracts and the contracts of the subcomponents that are part of the system in two separate steps: (1) by checking that all strong assumptions in a subcomponent that are not satisfied by the composition with other subcomponents are ensured by the strong assumption in the composite component contract, and (2) by checking that the composite component contract follows from the subcomponent contracts and the interconnections.

Sljivo [Sljivo et al. 2014] also propose a safety contracts for a component from the appliance of FLAR2SAF method. The behaviour of the individual component is expressed by a set of logical expressions (FPTC rules) that relate output failures (occurring on output ports) to combinations of input failures (occurring on input ports).

FLAR2SAF method can be performed by the following steps:

- Model the component architecture in a formal specification,
- Formally specify failure behaviour of a component in isolation,
- Translate the formal rules into corresponding safety contracts and attach system behaviour analysis based on individual component behaviour results as initial evidence,
- Support the contracts with additional V&V evidence and enrich the contract assumptions accordingly.

"A prudent man foresees the difficulties ahead and prepares for them; the simpleton goes blindly on and suffers the consequences." Proverbs 22:3

3

Standards Analysis

3.1 Introduction

This chapter describes the context of the functional safety standards for critical systems in the avionics, automotive and health domain from a composition perspective. We will analyse the different standards to find the similarities and differences when dealing with components assurance. The objective is to gather a good understanding to the level of complexity of the problem we are facing. We will focus on how compositional assurance is managed in the different domains.

Safety assurance and certification are amongst the most expensive and time-consuming tasks in the development of safety-critical embedded systems as it is evidenced in [Hawkins et al. 2013][Dodd Habli 2012]. Further, market trends strongly suggest that many future embedded systems will comprise heterogeneous, dynamic systems. As such, they will have to be built and assessed according to numerous standards and regulations. Current certification practices will be prohibitively costly to apply to this kind of embedded system. The certification process is carried out at the whole system level and components cannot be separately certified. In [RTI 2014], it is mentioned that for DO-178C, the standard which applies to software development in avionics domain, costs can range from $50 to $100 per executable line of code (ELOC), depending on the certification level. These are only the costs for creating the certification evidence and do not include the costs for designing and writing the code.

Rushby [Rushby 2007] defines certification objective as the intention to provide the stakeholders assurance that the system is sufficiently safe to operate avoiding unacceptable risk of adverse consequences. This assurance is based in three elements goals, evidences and argument that might appear explicitly or implicitly in the material provide for the assessment. The goals identify the hazardous event to be considered for avoidance or mitigation and the degree of risk considered acceptable. The evidence includes the results of activities such as analysis, reviews, validation and verification. Finally the arguments makes the connection on how the evidence support the goals already identified.

Standards in each of the domains analysed focus mainly on new developments. The system decomposition into parts that can be reused from previous projects is not clear in the standards proposed procedures. 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.

Reusing a project is difficult and even more when the context changes for example reusing across domain. Very few attempts have been made. Zeller [Zeller et al. 2014] proposed 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. SAFECER project proposed a use case where the focus of the reuse across domain was the tool qualification [SAFECER D5.4.1]. Their tool qualification proposal across domains is based on three pillars: (1) Cross-Domain Requirements spanning different standards, (2) Cross-domain development process according to the associated standards and their integrity levels, (3) Cross-domain tools, instantiated according to the associated standard.

When trying to analyse the commonalities of the different standards it will be beneficial to create a common framework so the particularities and commonalities of the standard get highlighted. In this chapter a common framework has been created in order to compare standards from the avionics, automotive and medical devices development functional safety standards.

This chapter is structured as follows. First, the different domains are described. The objective is to show the different composition and reuse approaches in each of the domains. In the next section a comparison framework is defined, this framework in then used to compare the commonalities and differences of each of the domains for assurance when dealing with component-based systems.

3.2 Fundamental concept of component

The different standards identified concepts for reuse which can be mapped to reusable components and need to fulfil some compliance requirements for a successful composition.

The first concept identified for the IEC 61508 is the compliant item. This standard, used on automation industries, is considered the “mother” of the functional safety standards analysed in chapter 3. The compliant item can be software or hardware but in both cases what the standard prescribes is the creation of a safety manual by the developers per compliant item. According to IEC 61508: “*The safety manual shall specify the functions of the compliant item. These may be used to support a safety function of a safety-related system or functions in a subsystem or element. The specification should clearly describe both the functions and the input and output interfaces.”* From this statement we can extract some requirements for the developers such as new artefacts that should be created, i.e., the safety manual, that should be released by the developers and these are pre-condition for the integrators to perform the integration phase.

One of the first problems encountered whenever we analyse standards from different domains is the definition of what composition means for each of them. Component, part or module, are concepts that might be used as synonyms, however, they have different meanings and connotation in each of the domains. Fig. 16 depicts a schematic on the decomposition of the concepts.

Standards do provide definitions related to components and composition, however, they do not agree on the naming or the description. Each domain has its own approach to system modularity and reuse. Component has become an overweight concept that could induce misunderstanding when different stakeholders discuss.

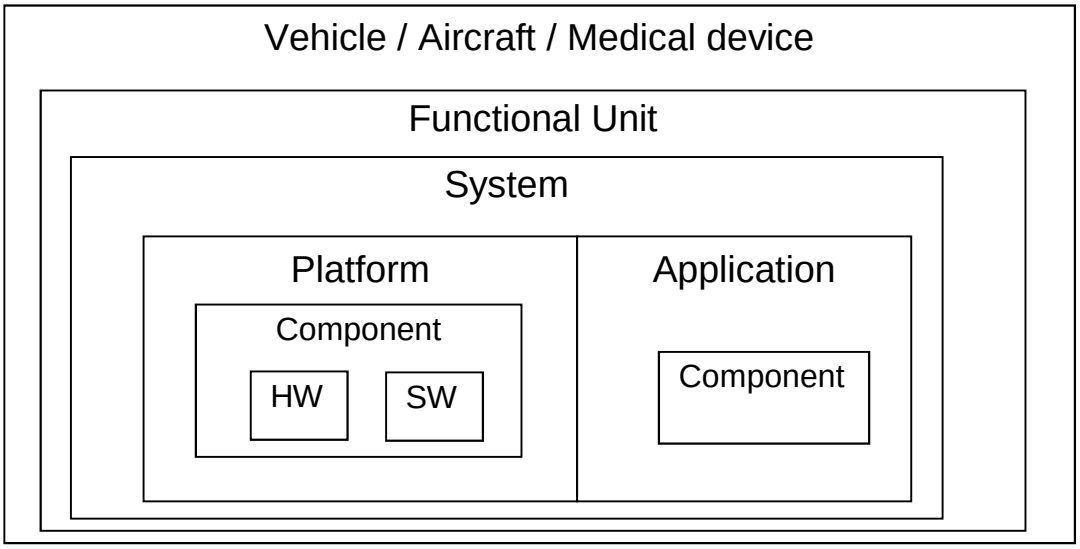


Fig. 16 Concepts decomposition

3.2.1 Avionics

The avionics domain is known for being a very standardized domain. In Fig. 17 the different actors involved on the aircraft manufacturing are shown [Chevrel 2011]. First the aircraft manufacturer agrees with the avionics authority of the country the type certificate. This certificate will include the first definition of the product with documents defining the aircraft characteristics. This is done at the very beginning of the design phase. The manufacturer then will make a contract with the different avionics system developers to contract the development of one or more systems of the aircraft and requires them their contribution for the airworthiness certification process. System development suppliers should also contact with the authority in order to get a Technical Standard Order (TSO) authorization to ensure their system is compliant with the avionics standards. Getting this authorization does not mean that the system will be certified. In fact, to install the system on the aircraft, the aircraft manufacturer has to discuss with the authority to get the installation authorisation. After the installation the complete aircraft goes through a safety assessment and it is after all the evaluation process that the aircraft is ready to get the airworthiness certification.

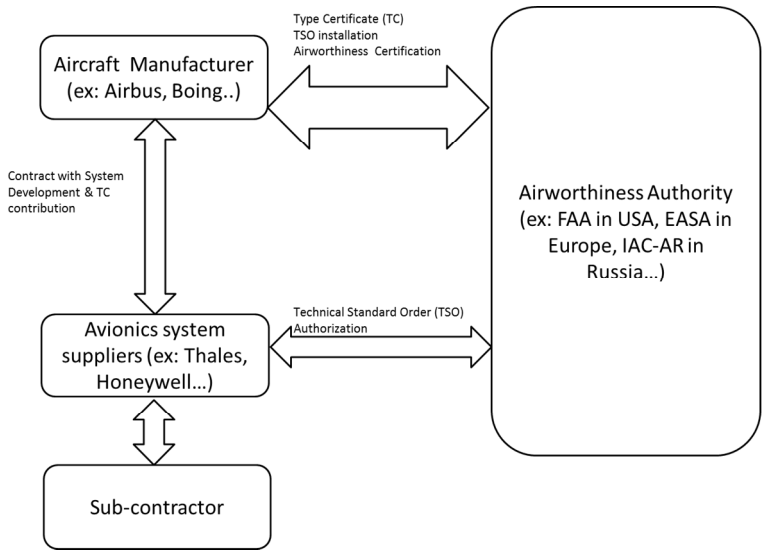


Fig. 17 Certification actors involved in avionics

In the avionics domain the DO-297 [DO-297] standard and the advisory circular AC 20-148 [AC 20-148] deal with reuse, each of the documents for a different scope. The AC 20-148 is the results of creating guidelines for the software component reuse while the DO-297 standard appears as a consequence of the move from federated architectures in avionics to IMA architecture (Integrated Modular Avionics). IMA is the term used for a distributed computing network aboard aircraft, which supports avionics applications of many different assurance levels, and it is designed for flexibility in configurations and modularity. It supports assurance evidence reuse to reduce effort required when reusing components in different systems. IMA technology has introduced the possibility to fragment the certification process into several tasks: (a) module and/or platform acceptance, (b) application acceptance (software and hardware), (c) IMA system acceptance (integration of multiple applications), (d) aircraft integration (e) change of modules or applications and (f) reuse of modules or applications.[DO-297]

The IMA platform architect role establishes a certification baseline about sizing hypothesis (memory, processor throughput), applicable (certification standards DO-254, DO-178), and functionality expected (e.g. API A653). The IMA platform architect also fixes the execution platform perimeter for the module supplier, including hardware (e.g., processing unit, IO units, and memory units) and software (OS, drivers, platform system functions, etc.). The module supplier provides what DO-297 calls the usage domain (characteristics and usage constraints). The module supplier provides qualification material for certification demonstrations as well. Finally, the IMA platform architect validates the module supplier's data and provides formal acceptation. Acceptance of a module can only be performed in the context of the aircraft or engine certification program or modification project.

In the avionics domain, federated systems are moving forward to more integrated modular avionics (IMA) architectures. IMA is the term used for a distributed computing

network aboard aircraft, which supports avionics applications of many different assurance levels, and is designed for flexibility in configurations and modularity. It supports assurance evidence re-use to reduce effort required when re-using components in different systems. The DO-297 standard [DO-297] proposes two concepts for compositions, applications and IMA platform modules.

In Fig. 18, extracted from the DO-297 standard, the IMA architecture is structured in reference to the different elements in which it is decomposed. On the one hand there are the avionics specific functionalities which are called the IMA applications and might be reused from one platform to another. On the other hand, the platform, the hardware and basic software in which the functions run are part of the architecture as well. Fig. 18 describes the incremental aspects that are related to the IMA concept. The IMA platform (certification domain #2) that is built from modules (certification domain #1) and associated configuration tools is in correlation with the IMA platform Usage Domain. The usage domain defines the set of rules and constraints that allow the customization of the IMA platform for a specific use while keeping the certification credits obtained for the platform.

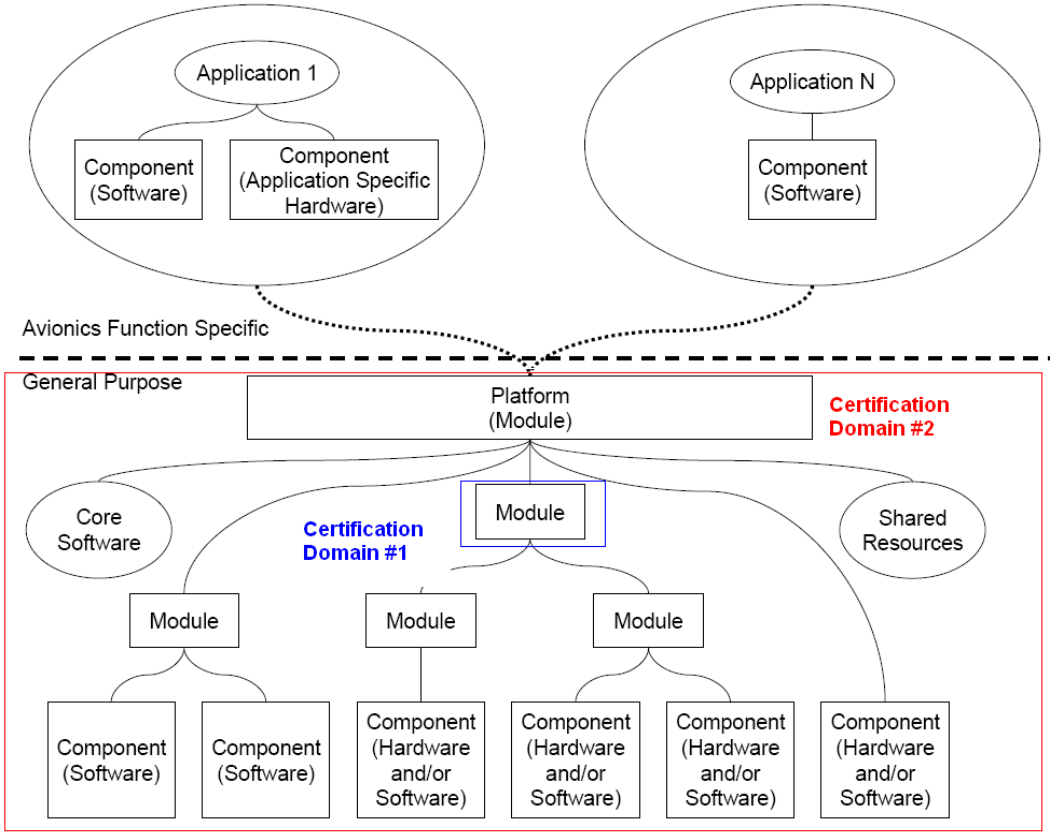


Fig. 18 Relationship of IMA elements and the incremental certification concept [Ruiz et al. 2012]

The objective of deploying an IMA architecture is to reduce the space, weight and power requirements by reducing the need of hardware. By complying with DO-297 the idea is to

reduce the cost of maintenance and certification as it allows the certification of the following building blocks: the IMA modules and the IMA application. It allows decomposing the certification process into the following activities:

- module and/or platform acceptance,
- application acceptance (software and hardware),
- IMA system acceptance (integration of multiple applications),
- aircraft integration (e) change of modules or applications,
- reuse of modules or applications.

IMA platform references a distributed computing network on-board aircraft, which allows us having support for avionics applications of many different assurance levels. This way, mixed-criticality applications can be running on the same platform without the need of certifying each of them to the highest assurance level. It is designed for flexibility in configurations and modularity but, at the same time, it ensures the isolation of the applications and the safety of the platform. It is thought for reusing applications from different target systems without increasing the certification costs.

However, applying DO-297 is not an easy task. Eveleens [Eveleens 2006] indicates as one of the challenges of reusing an IMA from a previous project the lack of sufficient support for dealing with changes made in existing IMA systems or when reusing design elements of an IMA. There is a need of justification in order to reuse pre-qualification documents due to the number of acceptance criteria, safety arguments and evidence that need to be considered in a new integration project.

Another hot topic for compositional certification is the incremental certification issue. In the AC 20-170 [AC 20-170] defines it as: “A process for obtaining credit toward approval and certification by accepting or finding that an IMA module, and/or off-aircraft IMA system complies with specific requirements. This incremental acceptance is divided into tasks. Credit granted for individual tasks contributes to the overall certification goal”. The tasks which conforms the incremental acceptance are:

- Module acceptance;
- Application Acceptance;
- IMA system acceptance;
- Aircraft Integration of IMA System (including Validation and Verification);
- Change;
- Reuse of modules or applications.

In Fig. 18 the elements for reuse on IMA architecture as Component (either software or application specific hardware), complete application, a module, a set of module or the complete platform are illustrated.

An IMA module, which is a module for the IMA platform, can work standalone or in a combination of more modules including core software, which manages resources in a sufficient manner to support at least one application. These applications running on the IMA platform are also eligible for reuse. For both, DO-297 prescribes a set of data that

should be available before the reuse takes place and some objectives that should be achieved. For example one of the objectives to fulfil is to ensure that life cycle data remains unchanged from what was previously accepted. Related to this, documents and data such as the MAP (module acceptance plan), PSAC (Plan for Software Aspects of Certification), PHAC (Plan for Hardware Aspects of Certification) should be available before the reuse is effectively done.

Fig. 19 based on the IMA certification process included in DO-297 standard includes the activities that should be done in each of the phases: (S) Specification, (D) Development, (I) Integration, (V) Validation and Verification, (Q) Compliance with standards and (C) Certification. Following the incremental certification guidelines the objective is to reduce the time required in task 3, the IMA system integration (off the aircraft) by reducing the effort needed in integration, Validation and Verification.

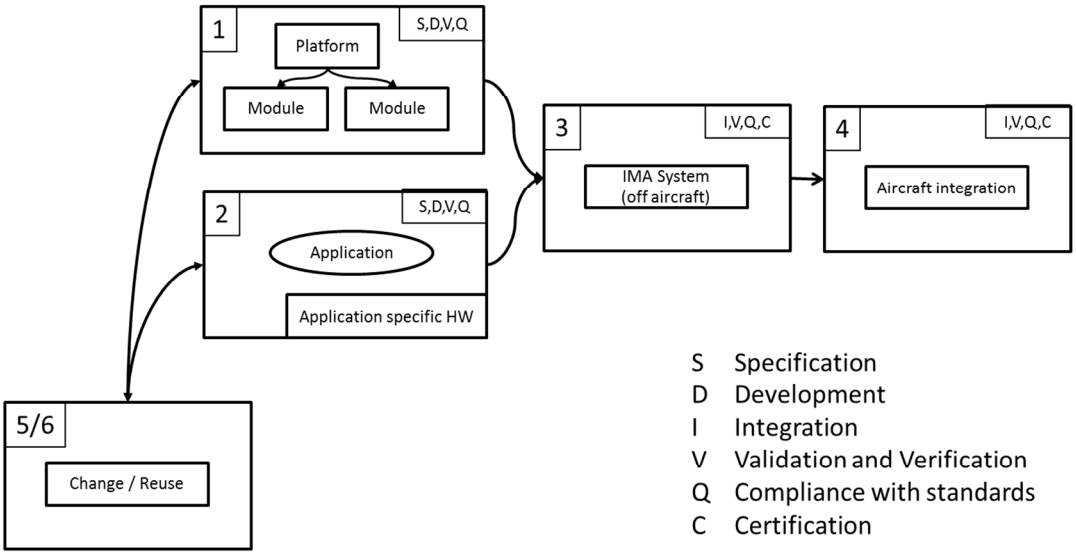


Fig. 19 IMA certification process

The advisory circular AC 20-148 is not a standard and consequently is not prescriptive; however, it is highly recommended applying it when using Reusable Software Components (RSC) in your system. According to the circular, a RSC "is the software, its supporting RTCA/DO-178B software life cycle data, and other supporting documentation being considered for reuse. The component designated for reuse may be any collection of software, such as libraries, operating systems, or specific system software functions". AC 20-148 distinguishes between what is the component from the developer view and from the integrator point of view.

The RSC needs to be developed having in mind that it is not going to be used in just once specific project but with reuse as an objective. This means that from the planning phase there should be some activities that should be performed and release a set of data to make reuse feasible. Some of these activities are referenced from the developer and from

the integrator perspective. For example, some of the activities suggested for the RSC developer to perform include:

- Prescribe activities to gain full credit for the installation
- Identify verification activities that integrator must repeat

Related to these the RSC integrator is required to perform the following activities:

- Activities prescribe by developer to gain full credit for the installation
- Repeat verification activities define by developer

These activities resemble pre conditions and post conditions that should be accomplished in order to make a satisfied reuse.

RSC is not the only concept in the avionics domain that takes reuse and composition into play. The DO-297 standard is based on the idea of reference architecture, the IMA architecture. The platform implementing this IMA architecture will be composed by modules and applications will run on top of this platform. The standard designates platform modules and applications to be reused on different projects.

Software development aspects are covered in DO-178 standard while hardware aspects are in DO-254 [DO-254]. These two standards describe development procedures and are referenced by the others.

Table 1 Scope of the avionics standards

Reference document	Scope
ARP 4754a	Assurance
ARP 4761	System
DO-297	Platform / Application
DO-178	Software
DO-254	Hardware
AC 20-148	Software components
AC 20-170	Incremental certification

In Table 2 some of the definitions used in the avionics standards regarding the compositional concepts are shown.

Table 2 Definition of compositional concepts according to avionic standards

Concept	Reference	Definition
Component	DO-297	A self-contained hardware or software part , database or combination thereof that maybe configuration controlled
	DO-178C DO-254	A self-contained part, combination of parts, subassemblies, or units that performs a distinct function of a system.

Concept	Reference	Definition
Item	ARP 4761	One or more hardware and/or software elements treated as a unit
Module	DO-297	A component or collection of components that may be accepted by themselves or in context of an IMA system
Application	DO-297	Software and/or application-specific hardware with a defined set of interfaces that when integrated with a platform(s) performs a function
Platform	DO-297	A module or group of modules, including core software, that manages resources in a manner sufficient to support at least one application
System	DO-297 DO-254	A collection of hardware and software components organized to accomplish a specific function or set of functions
COTS	DO-178	Commercially available applications sold by vendors through public catalogue listings. COTS software is not intended to be customized or enhanced- Contract-negotiated software developed for a specific application is not COTS software.
	DO-254	Component, integrated circuit or subsystem developed by a supplier for multiple customers, whose design and configuration is controlled by the supplier's or an industry specification.
Incremental certification	AC 20-170	A process for obtaining credit toward approval and certification by accepting or finding that an IMA module, and/or off-aircraft IMA system complies with specific requirements. This incremental acceptance is divided into tasks. Credit granted for individual tasks contributes to the overall certification goal

In avionics there are some properties that need to be checked for making reuse feasible. Some of these properties, that have their own section in the standards, include the robust partitioning and resource management. These properties should be reported to be ensured and results should appear on the validation and verification results report.

3.2.2 Automotive

The automotive domain is much less regulated. The standard ISO 26262 [ISO 26262] was published in November 2011 and is the reference for the functional safety in this domain. However, in the automotive industry, component driven development is a well-known strategy. Different concepts for compositional systems arise which differ on how they should be treated later on. Many suppliers take part on the development of a vehicle where different components and systems are integrated. There are three main concepts that cover the composition from different perspectives and are included on the standard:

- Hardware & Software qualified component;
- Safety Element out of Context (SEooC);
- Proven in use argument.

The three concepts share the aim of reusing the component together with the evidences which the show the compliance with the standard requirements.

In Fig. 20 and Fig. 21 taken from part 10 of the ISO 26262 standard we have a clear view of the parts in which an automotive system is decomposed, and the terms used in the standard to identify the scope of each of the elements. An item is always associated with a vehicle function and it item can be either a system or an element. Fig. 20 focuses on the relationship between the elements of a system. It clarifies the information from the standard perspective using the naming structure which appears in the standard.

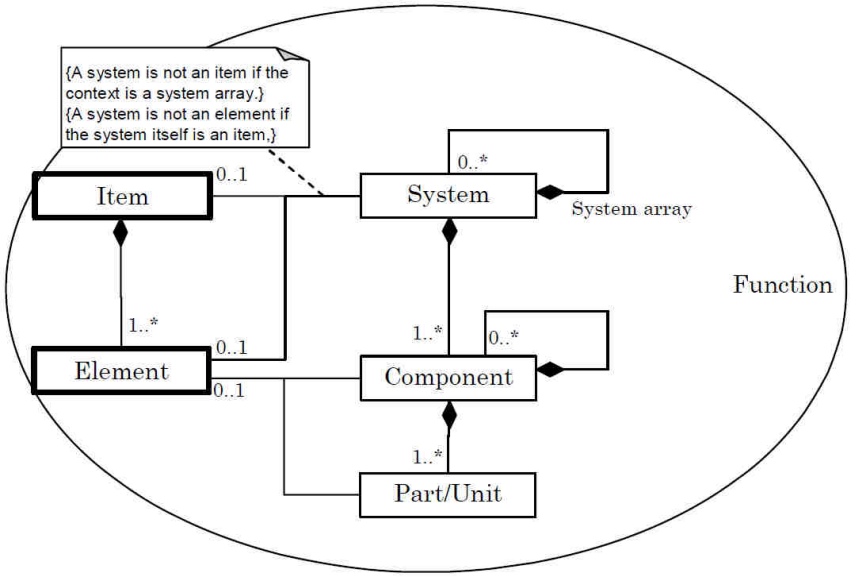


Fig. 20 Relationship of automotive concepts [ISO 26262-part 10]

Table 3 provides excerpts of definitions extracted from different standards that support the previous proposition. The definitions have been extracted from the part 1 of the standard which focuses on vocabulary.

Table 3 Definition of compositional concepts according to automotive standard ISO 26262

Concept	Reference	Definition
Component	ISO 26262	Non-system level element that is logically and technically separable and is comprised of more than one hardware part or more software units
Part/Unit	ISO 26262	Atomic level software component of the software architecture that can be subjected to standalone testing. When referring to hardware, then hardware which cannot be subdivided.
Element	ISO 26262	System or part of a system including components, hardware, software, hardware parts and software units
Item	ISO 26262	System or array of systems to implement a function at the vehicle level
System	ISO 26262	Set of elements that relates at least a sensor, a controller and an actuator with one another
SEooC	ISO 26262	A SEooC is a safety-related element which is not developed for a specific item.
Qualified Software/ Hardware component	ISO 26262	Pre-existing elements for an item which is not necessarily designed for reusability nor developed under ISO 26262

Fig. 21 is focused on the decomposition issue, i.e., how the electronics embedded systems in a vehicle are decomposed. In some situations it might be complex to clearly categorize a component as an item or as an element, the reason why is the overlapping; there are cases in which the element is an item, the boundaries are not fixed. That is the case of the SEooC which is not an item but an element and can be a system by itself.

Hardware and software qualified components appear on part 8 of the standard. The objective of these components is their reuse in items developed in compliance with ISO 26262 and with similar functionality and context.

A qualified hardware component is a piece of hardware which has not been developed according to ISO 26262. Some examples could be resistors, transistors, grey code decoder, sensors such as fuel pressure sensors. There is some discussion among suppliers whether sensors should follow ISO 26262.

A qualified software component is a piece of software to be reused on items with identical functionality. These components need to include source code, models, pre-compiled code, or compiled and linked software. Examples of these are the COTS software, or in-house components already in use.

Proven in use acceptance is only feasible when field data is available. The item or element for reuse which justifies the adequacy based on the data from previous produced vehicles which have incorporated the item. They shall have identical conditions of use as the previous one which should already be released and in operation. As the standard has only been released in November 2011, there are not many chances that elements are already in operation and with enough historic data to be shown as evidence when reused. The requirement for reuse about the conditions of use it is difficult to fulfil on different projects.

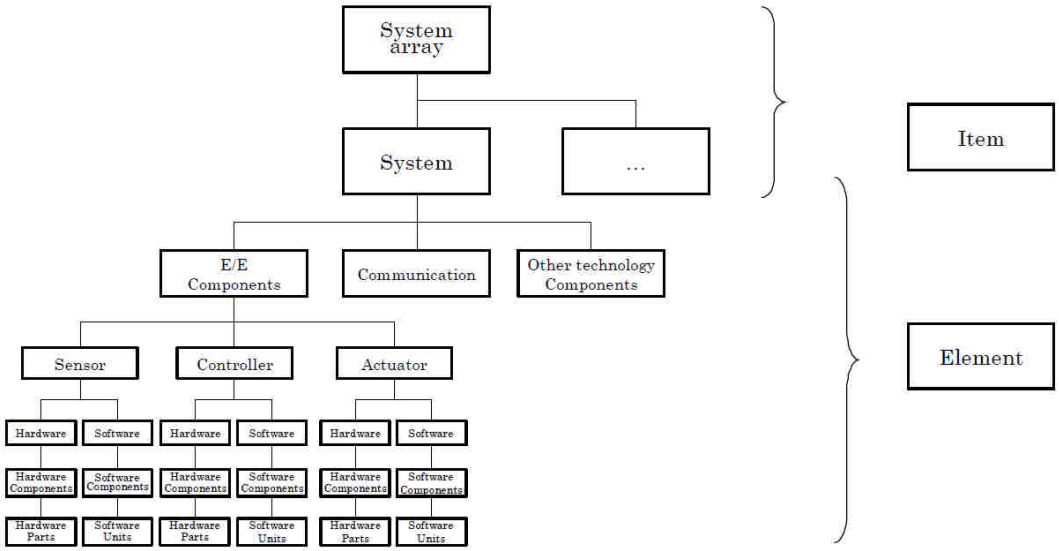


Fig. 21 Example of item dissolution taken from part 10 of the standard ISO 26262-10

Much more different is the SEooC concept which is described on the part 10 of the ISO 26262 standard. This concept is the one which share more commonalities with the IMA modules. The SEooC is developed from the beginning at the concept phase considering reuse. The reuse could be in different contexts where the final use could diverge. The possible contexts and system in which the component might end up being reused are assumed. The component might end be used for different purposes than the ones thought at development time, the possible contexts and end uses are assumed, in fact the assumptions that need to be define are of two types:

- External: related to the reference target (E/E architecture, system(s), environment, etc.)
- Internal: requirements and safety requirements, related to the application that are placed on the element by higher levels of design

In [Ruiz el at. 2013-1] we describe the activities for the SEooC compliance assessment (Fig. 22). SEooC does assumptions at the development phase on how it is going to be used and the environment in which it is going to be integrated. Once it is integrated with the item, these assumptions need to be validated. If assumptions are not compatible you should either change the item or the SEooC.

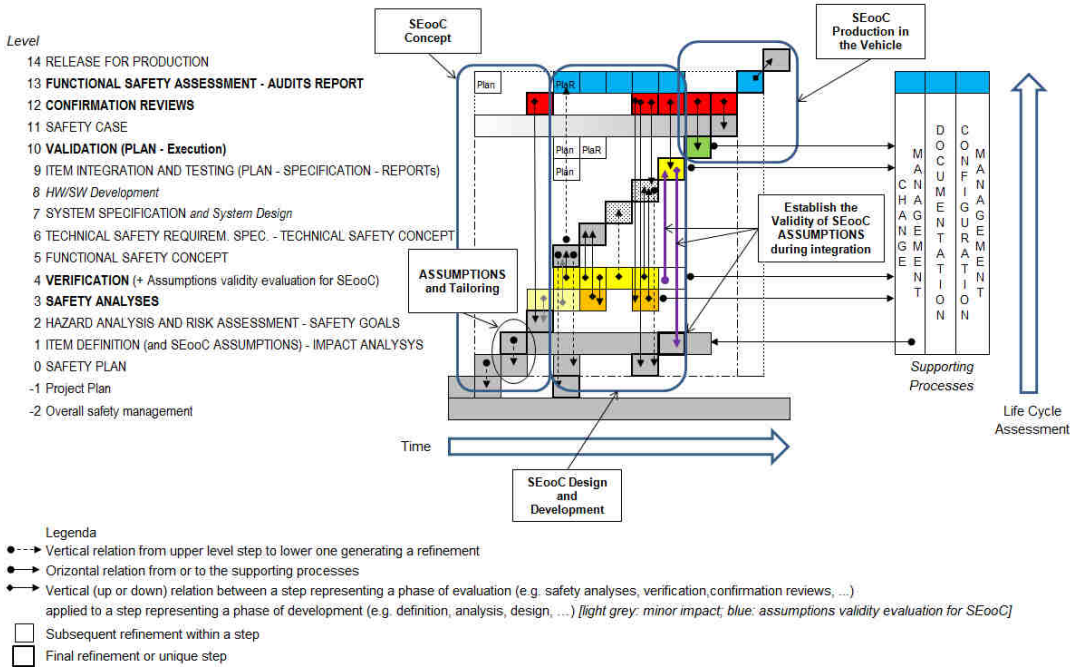


Fig. 22 An interpretation of being compliant with SEooC topics within a view of the progressing steps in a typical ISO 26262 workflow [Ruiz et al. 2013-1]

A SEooC is a safety-related element which is not developed for a specific item. The SEooC is a generic element that can be developed by an organization independently from the one that will be reusing the element in a specific context. It is developed under ISO 26262 and intended to be reusable. Examples of a SEooC can be:

1. A Hardware SEooC such as an ECU performing the inverter function that can be reused on: (a) the braking subsystem, (b) the clutch subsystem, (c) the accelerator subsystem
2. A Software SEooC like a communication software library that can easily be reused on: (a) Communication from the control system to the different systems, (b) Communication between two ECUS that are part of the same system.
3. A Software SEooC can also be an AUTOSAR [AUTOSAR]software module for controlling an electric motor which is meant to be integrated easily in a variety of potential car systems such as: (a) Electric power steering, (b) Dynamic steering, (c) Steer by wire, (d) Brake by wire, (e) Power sliding windows (f) Actuators for mirror adjustment

A system SEooC for example, the electronic parking system that is in charge of the management of the park pawl (mechanical engagement/disengagement) actuation. The parking system provides mechanical locking or unlocking of the transmission when the parking mode is selected (by the driver or automatically), avoiding unwanted movement of the vehicle when stopped. It can be reused on different vehicles.

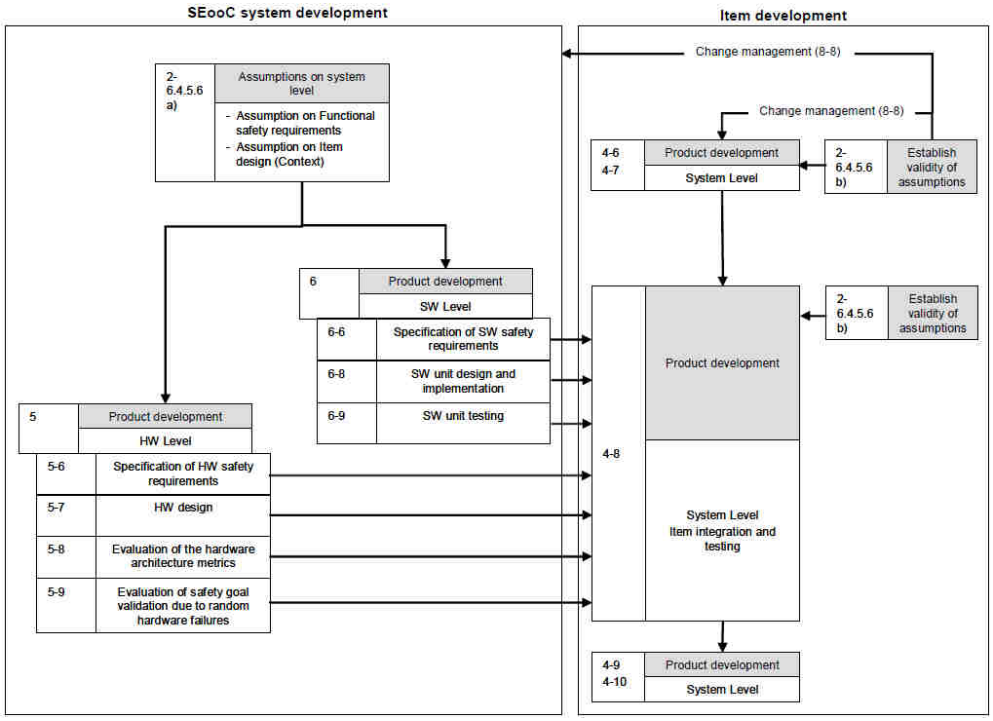


Fig. 23 SEooC System development

Fig. 24 depicts the process for SEooC development in accordance with ISO 26262 recommendations which appears in part 10. On the right side of the figure the specific activities that should be done at development time in a SEooC can be seen while on the left side the SEooC impacting on the item development is shown.

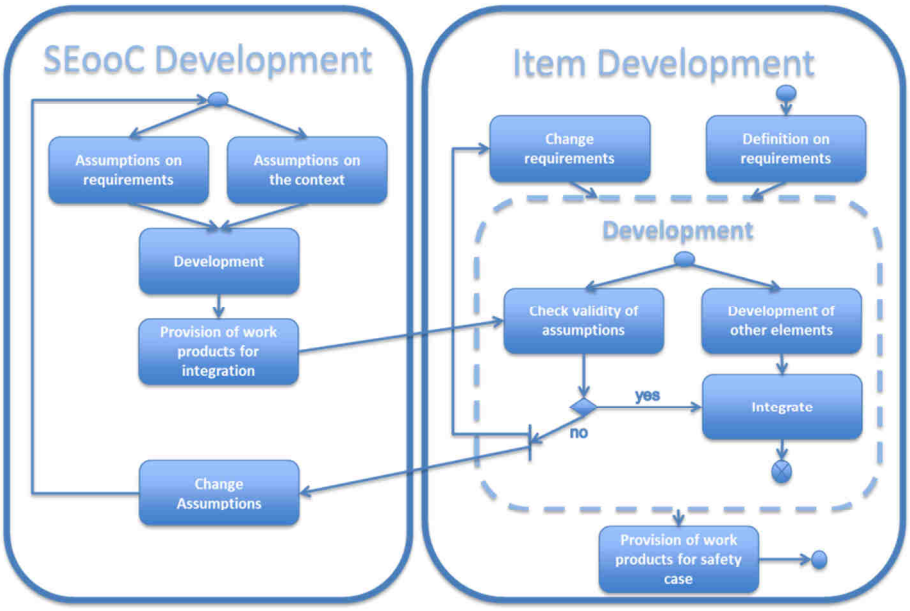


Fig. 24 SEooC development process

3.3.3 Medical devices

In the health domain where medical devices development is categorized, there is a standard ecosystem. Functional safety is split into three standards. The standard IEC 60601-1 is the generic standard. From this the 60601-1-XX are considered the collateral or horizontal standard. Part 2 standards, those following the naming series 60601-2-XX are defined for specific product types, such as the 60601-2-4 is specific for cardiac defibrillators. The horizontal standards, those names as 60601-1-XX to ensure certain aspect of the device (EMC; usability, alarms, environment…). Software related functional safety is handled in the IEC 62304 standard while risk management is done in the ISO 14971 standard.

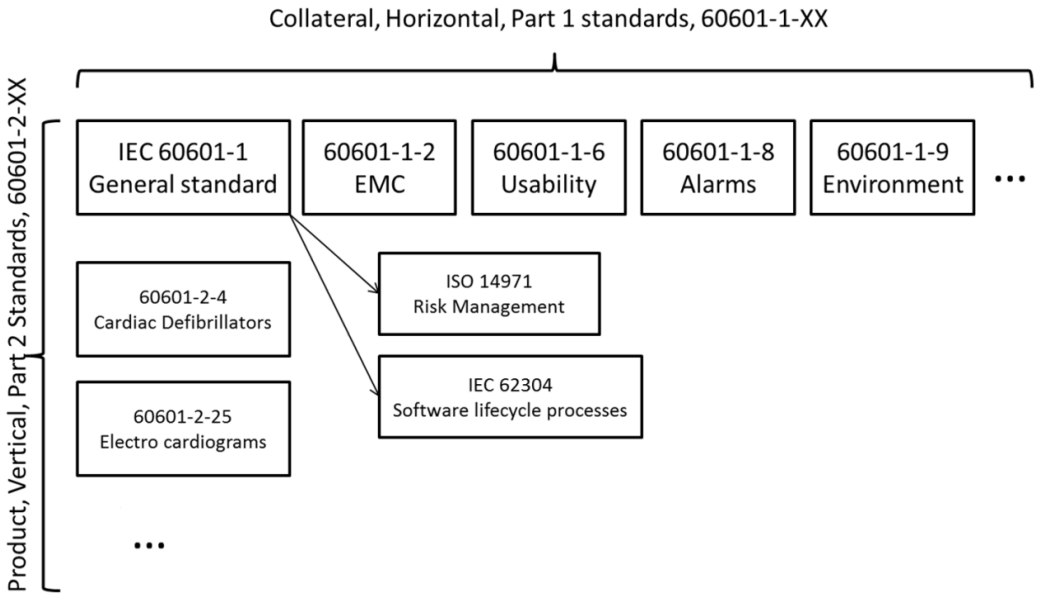


Fig. 25 Medical devices standards ecosystem

Regarding compositional assurance, a concept for SOUP (Software Of Unknown Precedence) is defined. It referrers to a software item that is already developed for which adequate records of the development processes are not available. Similar to automotive domain, the manufacturer of the medical device should tailor some of the activities in order to use a SOUP. From the planning phase, SOUP components should be identified. Manufacturer should also specify system hardware and software necessary to support the proper operation of the SOUP item. Hardware components are not so easy to identify as they are not mentioned on the generic standard, 60601-1. Hardware elements should be IEC 61508 compliant.

Table 4 Definitions of compositional concepts according to standard related to medical devices

Concept	Reference	Definition
Software item	IEC 62304 IEC 90003	Any identifiable part of a computer program

Concept	Reference	Definition
Software product	IEC 62304 IEC 12207	Set of computer programs, procedures, and possibly associated documentation and data
Software system	IEC 62304	Integrated collection of software items organized to accomplish a specific function or set of functions
Software unit	IEC 62304	Software item that is not subdivided into other items
SOUP	IEC 62304	Software of unknown provenance (SOUP). Software item that is already developed and generally available and that has not been developed for the purpose of being incorporated into the medical device (also known as "off-the-shelf" software") or software previously developed for which adequate records of the development processes are not available.
System	IEC 62304 IEC 12207	Integrated composite consisting of one or more of the processes, hardware, software, facilities, and people, that provides a capability to satisfy a stated need or objective.

3.3 Analysis on the guidelines

There have been a number of attempts to make a comparison between different standards from the different safety-critical-domains; however, there were not specifically focused on component compositions.

Ledinot in [Ledinot et al. 2012] made an analysis to different software safety standards: (DO-178/ED-12 for aeronautics, IEC 61508 for industry automation, ISO 26262 for automotive, IEC 60880 for nuclear, EN 50128 for railway and ECSS-Q-ST-80C for space) comparing how the Development Assurance Levels impact on the prescribed objectives, activities, methods or safety mechanisms to be implemented. They compare the probabilistic system safety levels to which the highest software development assurance levels are supposed to be compatible with. All the standards dismissed the notion of probabilistic software failure, as well as any probabilistic quantification of DAL (Development Assurance Level)-dependent likelihood of residual software fault. However, all the standards implicitly state that the various software development assurance levels are compatible, or consistent, with corresponding quantified system safety levels.

Blanquart [Blanquart et al. 2012] makes an analysis from the critical categories (Safety Integrity levels, Development Assurance levels, etc.) and highlights that all standard share the same fundamental concepts where critical categories are linked to the risk and effects of potential failures. The main divergence comes from acceptance barrier. Some domains

consider different categories for a few and many deaths, or consider damaging the environments or public or private property in same categories as injuries.

Safety standard guidelines on how to manage safety design in order to mitigate the possible risk has a direct impact on cost. The way to evidence the correct safety managements is by techniques that the standards associate with a specific Safety Integrity Level (SIL), DAL or risk classification [Machrouh et al. 2012] Machrouh also mentioned that "Defining the commonalities between safety standards in various domains allows one to reduce the development cost of the critical-embedded systems by mutualising the developments by reuse of components".

Papadopoulos and McDermid in [Papadopoulos McDermid 1999] define a reference structure for the comparative review of standards. The structure is based on five principal dimensions of the certification problem: (1) Requirements for system development and safety processes, (2) Method for establishing the system Safety Requirements, (3) Definition, treatment and allocation of development assurance levels, (4) Requirements on techniques for component specification, development and verification and (5) Requirements on the content and structure of the safety case. They identified significant differences "in the recommended component specification, development and verification techniques, at the detailed level".

None of these comparisons took in consideration the perspective of component assurance and reuse requirements. The medical device related standards do not appear on the previous comparison. For that reason we have tried to provide a brief analysis on the avionics, automotive and health domains. These domains provide a good view of the different standards situation. All these domains consider the IEC 61508 [IEC 61508] as the mother standard in which they have based their own domain specific standards. All these domains differ on their approach to functional safety. Avionics standards trust on process-based standards, automotive applies a goal-based standard for functional safety while standards for medical devices development can be categorized as product-based. These provide us a good view of safety standards and have been selected for the comparison and later on to be the domains in which the case studies have been developed.

In Fig. 26, we have tried to describe the scope of the different standards from the avionics, automotive and health domains.

In order to make a more systematic comparison we have defined a comparison framework using the GQM (Goal Question Metrics Technique). "Writing goals allowed us to focus on important issues. Defining questions allowed us to make goals more specific and suggested the metrics that were relevant to the goals." [Solingen Berghout 1999]. Table 5 depicts the analysis done for the framework creation

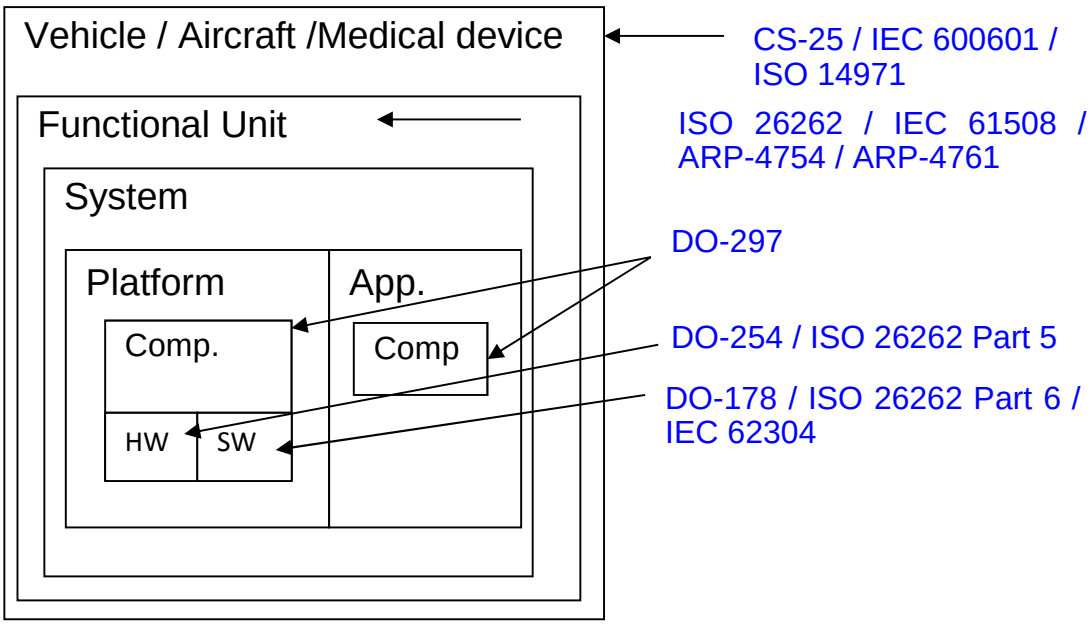


Fig. 26 Standards scope in relation to decomposition concepts

Table 5 GQM definition for standards analysis

Goals	Questions	Metrics
Scope	Which is the scope of the elements to be reused?	Is reuse on the system allowed?
		Is reuse on a system level allowed?
		Is reuse of SW systems allowed?
		Is reuse of SW components allowed?
		Is reuse of HW systems allowed?
		Is reuse of HW components allowed?
Context	Which is the scope of reuse permitted?	Is the reuse done on same context or different context that previous one?
Complexity level/ Integration level		Number of activities impacted by the reuse
		Number of artefacts impacted by the reuse
		Number of properties to be verified after the integration
Composition requirements	Is the composition included on the standard?	Number of guidelines to comply with in order to perform a reuse

For the analysis we have taken into account the following standards and guidelines: ISO 26262, DO-297, DO-178, DO-254, ARP 4754, ARP4761, AC 20-148, AC 20-170, IEC 60601, ISO 14971 and IEC 62304. In the following table the results from this comparative studio is shown.

Table 6 Results of the GQM to the standards analysis

Metrics	Avionics	Automotive	Medical Devices
Is reuse on system level (vehicle...) allowed?	IMA application	Proven in use	n.a.
Is reuse on a system level allowed?	n.a.	System SEooC Proven in use	n.a.
Is reuse of SW systems allowed?	IMA software module	SW SEooC	Software product
Is reuse of SW components allowed?	RSC (reusable software component)	Qualified SW Software SEooC	SOUP
Is reuse of HW systems allowed?		HW SEooC	n.a.
Is reuse of HW components allowed?	Just the FAA accepts the hardware component reused on an IMA platform on its ETSI 504 order.	HW SEooC Qualified HW	Hardware compliant item (according to IEC 61508)
Is the reuse done on same context or different context that previous one?	Different context are accepted. Using the concept of RSC (reusable software component) where analysis should be done to ensure the behaviour. IMA modules accept to be used on different contexts if previously the have been defined on parameters which can be configured.	The reuse can be done on the same context by applying the concepts of qualified Software or proven in use. Different Context reuse is also accepted, this time the concept of SEooC is applied	Different context are accepted. The concept SOUP (Software Of Unknown Provenance) permits the use of software from other context.

Metrics	Avionics	Automotive	Medical Devices
Number of activities impacted by the reuse	31. We have analysed the activities mentioned on DO-297 and on AC 20-148	9. We have analysed the activities mentioned on ISO 2626	14. We have analysed the activities mentioned on IEC 62304
	See the annex A for the analysis of activities mentioned on the standard that should be ensured for reuse.		
Number of artefacts impacted by the reuse	37. We have analysed the artefacts mentioned on DO-297 and on AC 20-148	45 (SEooC), 6 (Qualified software component), 3 (qualified hardware component). We have analysed the activities mentioned on ISO 2626 and depending on the concept we will apply for reuse the number of artefacts differ.	As the artefacts are not defined by the standard, this metric do not apply
	See the annex A for the analysis of the properties mentioned on the standard that should be ensured for reuse.		
Number of properties to be verified after the integration	36.	30	24
	See the annex A for the analysis of the properties mentioned on the standard that should be ensured for reuse.		
Number of guidelines to comply with in order to perform a reuse	AC 20-148 for software DO-297 for IMA TSI for hardware	Part 10 of ISO 26262 for guidelines on qualified HW&SW and for SEooC	Guidance for Industry, FDA Reviewers and Compliance on Off-The-Shelf Software Use in Medical Devices

3.3.1 Common features

Taking all the previous information in account, we have developed a harmonized criterion for comparison and management. Guidelines and standards give us the notions about the information we need to manage with respect to assurance, those are the concepts that a components need to deal with. Details on the results for the analysis can be consulted in annex A. When analysing the material we noticed that the data required for assurance can be classified in three main categories which forces the proposal for compositional assurance to include:

- Artefacts: referring to the data required by an authority or similar when doing the safety assessment
- Properties: these are characteristics that need to ensure before and after the integration in order to confirm that there are no concerns or an emerging unknown behaviour.
- Processes: refers to activities that should be performed in order to prepare the reuse and after the reuse itself in order to comply with the standards requirements.

All the tables shown in annex A can be seen as viewpoints of the component, defining a subset of information which gives useful information. However, the tables should not be understood by themselves but in combination. They are linked, each of them impacting the other, artefacts, properties and processes are connected.

One of the core concepts in assurance standards is the notion of explicit and traceable requirements. In the avionics domain, specifically for software two important documents are defined in the DO-178 standard: the Software Requirements Standards and the Software Requirements Data artefacts.

According to DO-178 clause 11.6: "Software Requirements Standards define the methods, rules, and tools to be used to develop the high-level requirements". Software Requirements Standards should include:

a) The methods to be used for developing software requirements, such as structured methods.
b) Notations to be used to express requirements, such as data flow diagrams and formal specification languages.
c) Constraints on the use of the tools used for requirements development.
d) The method to be used to provide derived requirements to the system processes.

Clause 11.9 defines the information the software requirements data shall contain: "Software Requirements Data is a definition of the high-level requirements including the derived requirements. This data should include:

a. Description of the allocation of system requirements to software, with attention to safety-related requirements and potential failure conditions.
b. Functional and operational requirements under each mode of operation.
c. Performance criteria, for example, precision and accuracy.
d. Timing requirements and constraints.
e. Memory size constraints.
f. Hardware and software interfaces, for example, protocols, formats, frequency of inputs, and frequency of outputs.
g. Failure detection and safety monitoring requirements.
h. Partitioning requirements allocated to software, how the partitioned software components interact with each other, and the software level(s) of each partition."

The standard also connects these two artefacts in what is defined as the "Software Requirements Process Activities" in clause 5.1.2. In that clause it is mentioned: Inputs to the software requirements process include the system requirements, the hardware interface and system architecture (if not included in the requirements) from the system life cycle processes, and the Software Development Plan and the Software Requirements Standards from the software planning process. When the planned transition criteria have been satisfied, these inputs are used to develop the high-level requirements. The primary output of this process is the Software Requirements Data."

We have here two artefacts: the Software Requirements Standards and the Software Requirements Data. The first one is the input of the activity "Software Requirements Process Activities" while the second one is the output of the mentioned activity. This is not an activity specific for the reuse but related to the development itself. When looking to which properties to these artefacts and processes are involved; we need to pay attention to the following:

- Interface specification
- Intended use description
- Safety features
- Safety related requirements
- Failure categories
- Operational specifications
- Architecture & design features
- Performance specifications

The information is linked one another and although the data is required in different formats depending on the stakeholders, all is necessary in order to be able to compose the argumentation for assurance compliance.

Another example that can be found in avionics where the link is cleared that previous one and which is essential for component composition is the so called Reusable Software Component (RSC) artefact data Sheet. This artefact is the key for selecting one component or another to be reuse on a new project. It is specified on AC 20-148 clause 6i. "… This data sheet must concisely summarize:

- RSC functions;
- Limitations;
- Analysis of potential interface safety concerns;
- Assumptions;
- Configuration;
- Supporting data;
- Open problem reports;
- Software characteristics; and
- Other relevant information that supports the integrator's or applicant's use of the RSC."

AC 20-148 also mentions that the RSC developer shall "produce an analysis of the RSC's behaviour that could adversely affect the users' implementation (for example, vulnerabilities, partitioning requirements, hardware failure effects, requirements for redundancy, data latency, and design constraints for correct RSC operation). The analysis may support the integrator's or applicant's safety analysis." This artefact identifies many of the properties for reuse we have included on annex A.

The settable parameters specification property is related with the behaviour as it is the selection of a parameter what impact on a specific behaviour or another. Also all the properties which might be an issue for reuse such as: Timing, Memory Usage. Resource usage, Resource items, Data coupling, Partitioning, Protection, Deactivated code, Traceability or Robustness are part of the content of this artefact. The value they have is not enough; the method used for calculating it is needed as well. Many activities are related to define, evaluate, validate and verify those properties at development time but also at the integration time. The developer should proceed with:

- Analysis of any potential functional, operational, performance and safety effects
- Analysis of all interfaces
- Analysis of all settable parameters

While the integrator also needs to perform activities related such as:

- Validate and verify the throughput, timing, memory usage, resource usage, and other resource items
- Open problem reports on the RSC and analysis of any potential functional, operational, performance and safety effects
- Analysis of data coupling and control coupling of the RSC
- Retest where new setting or parameters may affect the requirements, code, function, performance, or protection features
- Validate the assumptions made by RSC's developer

This is not only an avionics case, it also affects to the automotive domain. Focusing on how requirements are handled on the automotive domain where evidence that the SW component complies with its requirements is required. The objective of this document is to:

- Show requirement coverage. This is done by showing the requirements coverage on the different Software verification Plan, Software verification specification and Software verification report
- Cover both normal operating conditions and behaviour in case of failure
- Results from known errors to show that they do not lead to the violation of safety requirements

When looking at the content we can easily map with some of the properties we have identified before, the safety requirements, the maximum ASIL of any safety requirement, the requirements coverage, and requirements for testing equipment, the known anomalies or the failure modes. The activities proposed in the standard related to these properties and artefacts are the software component specification or verification activities for both normal operating conditions and behaviour in the case of failure and known error verification.

In the medical device domain according to the guidance for off the self (OTS) software use on medical device, the component should provide the basic documentation. This documentation should answer the following questions:

"1. What is it? - For each component of OTS Software used, specify the following:

- Title and Manufacturer of the OTS Software.
- Version Level, Release Date, Patch Number and Upgrade Designation as appropriate.
- Any OTS Software documentation that will be provided to the end user.
- Why is this OTS Software appropriate for this medical device?
- What are the expected design limitations of the OTS Software?

2. What are the Computer System Specifications for the OTS Software? - For what configuration will the OTS software be validated? Specify the following:

- Hardware specifications: processor (manufacturer, speed, and features), RAM (memory size), hard disk size, other storage, communications, display, etc.
- Software specifications: operating system, drivers, utilities, etc. The software requirements specification (SRS) listing for each item should contain the name (e.g., Windows 95, Excel, Sun OS, etc.), specific version levels (e.g., 4.1, 5.0, etc.) and a complete list of any patches that have been provided by the OTS Software manufacturer.

3. How appropriate actions taken by the End User are assured?

- What aspects of the OTS Software and system can (and/or must) be installed/configured?
- What steps are permitted (or must be taken) to install and/or configure the product?
- How often will the configuration need to be changed?
- What education and training are suggested or required for the user of the OTS Software?
- What measures have been designed into the medical device to prevent the operation of any non-specified OTS Software, e.g., word processors, games? Operation of non-specified OTS Software may be prevented by system design, preventive measures, or labeling. Introduction may be prevented by disabling input (floppy disk, CD, tape drives, modems).

4. What does the OTS Software do? – What function does the OTS software provide in this device? Specify the following:

- What is the OTS Software intended to do? The sponsor's design documentation should specify exactly which OTS components will be included in the design of the medical device. Specify to what extent OTS Software is involved in error control and messaging in device error control.
- What are the links with other software including software outside the medical device (not reviewed as part of this or another application)? The links to outside software should be completely defined for each medical device/module. The design documentation should include a complete description of the linkage between the medical device software and any outside software (e.g., networks).

5. How do you know it works? – Based on the Level of Concern:

- Describe testing, verification and validation of the OTS Software and ensure it is appropriate for the device hazards associated with the OTS software.
- Provide the results of the testing.
- Is there a current list of OTS Software problems (bugs) and access to updates?

6. How will you keep track of (control) the OTS Software? - An appropriate plan should answer the following questions:

- What measures have been designed into the medical device to prevent the introduction of incorrect versions? On start-up, ideally, the medical device should check to verify that all software is the correct title, version level and configuration. If the correct software is not loaded, the medical device should warn the operator and shut down to a safe state.
- How will you maintain the OTS Software configuration?
- Where and how will you store the OTS Software?
- How will you ensure proper installation of the OTS Software?
- How will you ensure proper maintenance and life cycle support for the OTS Software?"

The data for compliance is wide in the sense that many topics and areas are affected by composition assurance but also that they are closely related. Other models have been focusing on one on the areas presented here and the last example the behaviour properties are a good example on this. This has been tackle at technical level by using reference architecture such as IMA or AUTOSAR and EAST-ADL or MARTE [MARTE 1.1] do support the specification of these properties. Other standards like BPMN [BPMN 2.0] or EPF (Eclipse Process Framework) provide the functionality to express the information related to process. However, these models lack in the work of relating all of this for the compliance purpose in general and at composition assurance specifically.

3.4 Overview of the proposed approach

After the standards analysis, we present the main elements underpinning the approach presented in this thesis. This thesis has been partially framed under the European OPENCOSS project.

In Fig. 27, the block on the top left corner represents a component-based system where a component is missing. This system will be the target system in which a component will be integrated. When we analyse the integration of the target system plus the component assurance data, we result in a system where there is still missing information for a complete system assurance analysis. At the bottom of the figure is shown the approach followed in this thesis where we do not only include component assurance data but also integration assurance data for a complete system assurance analysis.

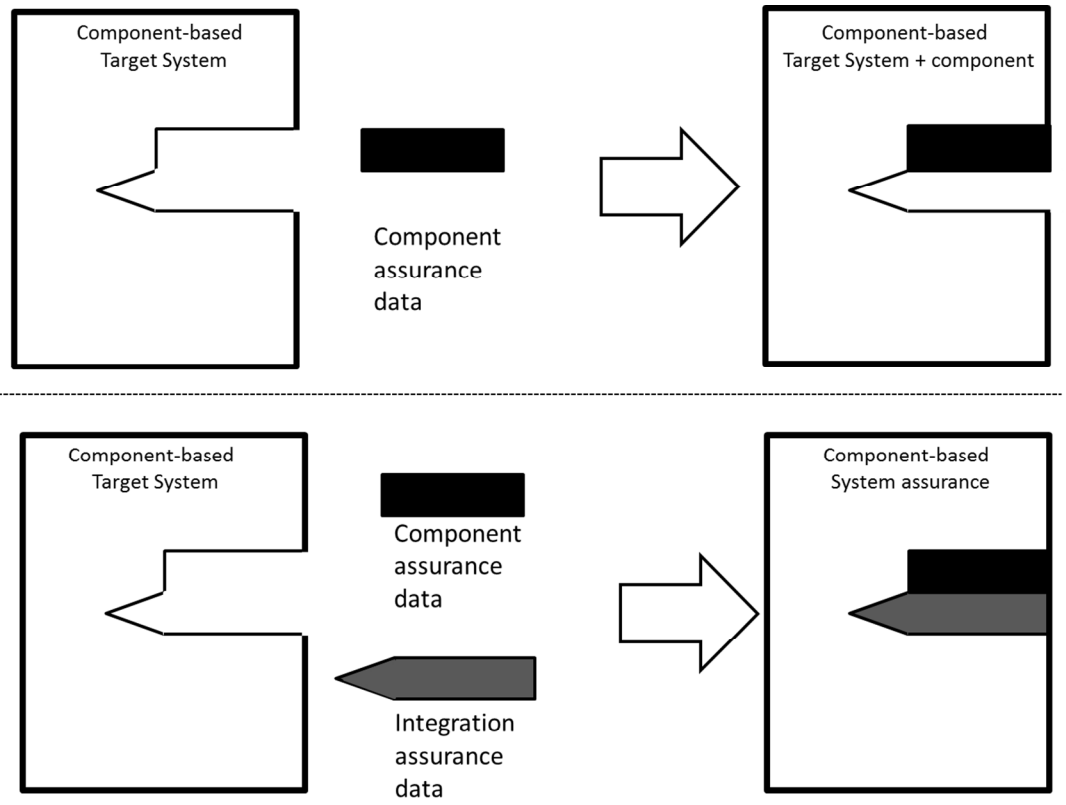


Fig. 27 Compositional assurance description

As a result of this approach we get an incremental assurance process as it is shown in Fig. 28**¡Error! No se encuentra el origen de la referencia.**. The platform or system is being developed independently from the component development. The objective is to provide enough assurance information to the integration phase, so this phase is reduce in time and effort before it is ready for a complete system assurance analysis.

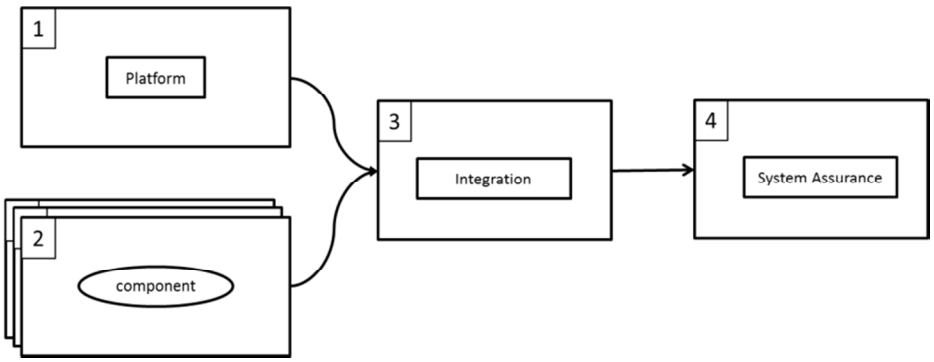


Fig. 28 Incremental assurance process

We proposed here a common high level process for the compliance process for component-based systems that will be operating in safety-critical domains. This process is shown in Fig. 29.

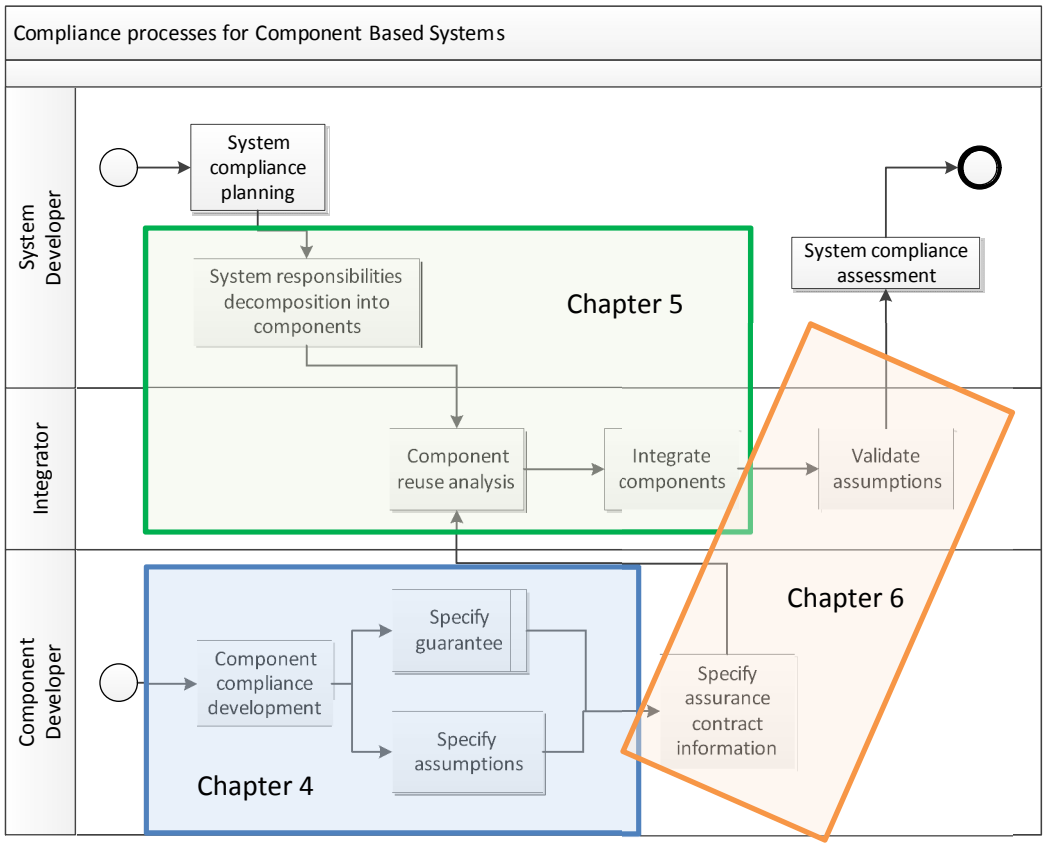


Fig. 29 Compliance processes for component-based systems

The blue square highlighting activities done by the component developer are described and decomposed in detail in chapter 4. The green square highlighting activities done by the component integrator are described and decomposed in detail in chapter 5. Finally, the orange square remarks the activities that will be described in chapter 6.

"If you talk to a man in a language he understands, that goes to his head. If you talk to him in his language, that goes to his heart." – Nelson Mandela

4

Assurance Modelling

4.1 Introduction

This section focuses on the assurance problems that have been mentioned on Chapter 1 and the needs for modelling that have been identified in the standard analysis described in Chapter 3. In the previous section, the needs regarding compliance with the objectives of standards have been highlighted. Furthermore, the heterogeneous objectives and information that should be available in order to make a system compliant within one or more standards according to the domain it is going to be used have been mentioned. The information required to be managed is complex. In different domains we have different approaches for safety compliance and for composition assurance; however, all are ruled by standards.

In the previous chapter, we have analysed the information contained in the standards. More concretely, we have analysed how standards and assurance processes treat components of a system in the different domains. We have also detailed the differences among each of the domains along with the assurance lifecycle and how it connects to the final product development. We have finally identified the commonalities and points of agreement between standards. These commonalities and the capability to specify the variances will be the basis of this section.

In this section, we propose modelling as a means for assurance management. Having a common approach for standards modelling will benefit different stakeholders by providing a framework for comparison. This framework will support comparison among distinct projects in the same domain but as a secondary goal, it also provides stakeholders from the different domains with a common language to relate the various concepts for assurance. It can also improve communication among stakeholders by reducing misunderstandings due to differences in interpretation. Stakeholders from different phases of the development could have different views as to the level of detail required for a specific development artefact, e.g. a safety plan. Having a common framework can support the understanding of the compliance requirements for each of the stakeholders that collaborate on demonstrating compliance of the system at all levels of integration.

This chapter is organized as follows. First the challenges section introduces the requirements from the standards regarding assurance. We will identify the issues we identified and that need to be analysed in order to provide a common modelling

framework. Following this, there is a literature review on existing approaches in modelling assurance. The next section concerns how standards are able to be modelled with a common modelling language for different domain standards. Following this we talk about the capability to model project specific requirements by tailoring or interpreting the requirements of standards. Then we describe how argumentation of the project can be explicitly recorded, and finally how evidence artefacts should be managed.

As in previous s chapters, we use a process view of the component-based development to place the contributions of this thesis. In Fig. 30 we have highlighted the activities that are covered in this chapter. In the previous chapter, we have described compliance processes for component-based systems from the high-level perspective. In this chapter, we focus on components compliance along the component development.

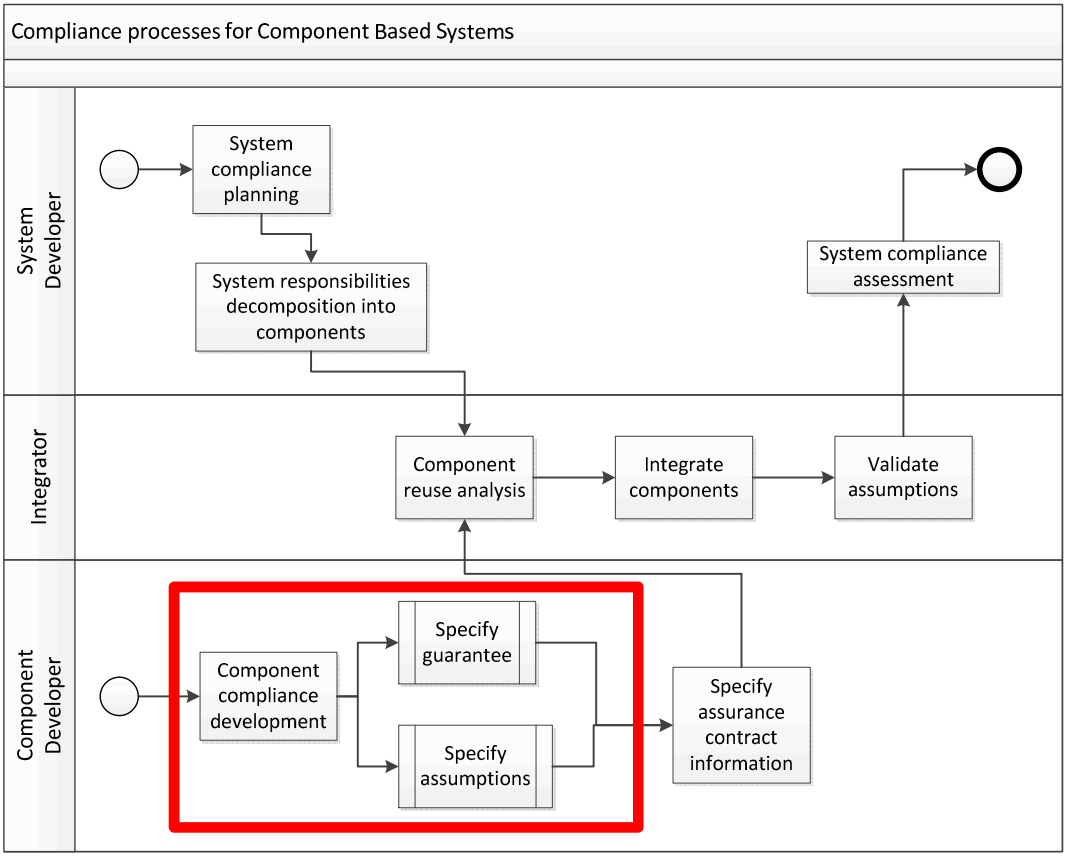


Fig. 30 Scope for chapter 4 in the compliance processes for component-based systems

Note: Whilst much of this chapter is joint work with the OPENCOSS project team, design rationales, examples, and methodological contributions in this chapter have been entirely defined by the thesis author. The remaining thesis contributions are not a part of OPENCOSS.

4.2 Challenges

Component-Based Software Engineering (CBSE) is seen as a common and well-known strategy for dealing with complex systems. As systems grow in complexity, so does the trend in using components.

However, when assuring a component-based system, issues arise. One of the first questions we identified concerns the standard the component should comply with. As mentioned in the previous chapter 3, sometimes it is not simply mandatory standards which apply but also guidance material, such as the advisory circular AC 20-148 in the avionics domain for reusable software component. This guidance references the need to comply with parts of the DO-178C standard. If the component will later be part of an IMA application or part of an IMA module, then the DO-297 standard should also be applied. There is not a systematic way to show compliance to the set of standards the component has fulfilled.

Standards often define objectives that cover the entire system lifecycle and the development of complete systems. Component developments can find it difficult to show compliance with all of these objectives. Components developers need to find a way to show in a clear and unambiguous way how far they have gone in standard compliance.

Component developers also need to produce artefacts along with their development and assessment lifecycle. These artefacts might be later on be used when demonstrated compliance within the integrated system. It is not clearly stated in the standards the level of details these artefacts should cover, or as a result of what activity they are created or what claim they are supposed to support.

Each of the components should have their own assurance case arguing about the safety of the component product, the compliance with standards and the level of confidence in the demonstration of safety. However, some of the claims presented will be based on the assumptions made concerning the characteristics of other components, like the operational environment or a specific property of the overall system such as the assumptions made during the SEooC development. The component makes assumptions about claims supported by other components and about evidence from other components that are used to support claims done at component level. Inside the assurance case the component developer may analyse many properties of the component, but chose to only reveal certain claims about the component to those using the component. In this case, we can consider the component assurance case as a black box with assumptions and guarantees, which can be utilised by external stakeholders.

An overall system could be developed using components developed by different suppliers using different assurance frameworks. It is important from an industry point of view that assurance information about components is provided in a standardised, recognisable format.

Regarding the assurance case, developers need a framework to help them structure their claims about their components, and the assumptions that they have made so that when

the component is reused the necessary safety assurance information for integration is available.

These challenges are not a problem just for component-based systems but for safety-critical systems in general. Larrucea identifies [Larrucea et al. 2013] the following challenges in safety-critical systems in relation with assurance cases and evidence approaches:

- **Unawareness of the certification process**. The lack of awareness on the certification aspects is a frequent problem in the current practice, in large part arising due to poor visibility into the architecture of systems, their design rationale, how components were verified and integrated, and finally how the system components and the system as a whole were certified. This is related to the lack of information a developer has when dealing to which standard the components will work into.
- **Data exists in many places, with different formats, multiple copies and versions**. Usually, engineers submit paper-based reports and do not know where the reports go and are unable to follow up. Quality and safety managers assess and classify information. Excel and Word documents are often exchanged, of which multiple copies and versions exist. The detail of information requested for the component evidence is linked to this problem.
- **Difficulties in interpretations of argumentation**. Determining the degree of compliance with specified standards or practices for the different safety-critical market and technological domains is a challenging task. There are a variety of definitions of evidence, and how to evaluate it or derive it in regard the technology used, which makes cross-acceptance difficult.

We [Espinoza et al. 2011] identified several factors that impact on the safety-critical systems such as

- Lack of precision and large variety of certification requirements
- Lack of composable/modular view for certification
- High and non-measured costs for (re)certification
- Lack of openness to innovation and new approaches

4.3 Existing Assurance Modelling Approaches in the Literature

In chapter 3 Standards Analysis, we have identified the information that should be modelled for assurance regarding components. Components assurance modelling shall treat processes, artefacts and properties. Previous approaches in the literature have been mainly created not for components but for system assurance. It is important to highlight their coverage for:

- handling different standards,
- manage information about the process followed,
- trace the artefacts evolution from the planning, during the development and after the integration,

- treat properties about the components and its behaviour and context,
- and if there is tool support.

4.3.1 Standards modelling

In [Arana et al. 2011] a Software Engineering Process for Certification Metamodel is proposed. They created a general certification "language" by means of a structured semi-formal meta-model, which will act like a template for certification requirements specification.

It includes information about standard regulations however, the mechanism for showing compliance is based in checklists without any structured justification. Also, product safety and evidential artefacts are not traced during its evolutions.

They still need to consolidate a set of concepts for compliance management. This approach still lacks for reuse strategies (from standard to standard, product upgrade, assessment scope). As it has just been mentioned, there is no integration with evidential repository or with process-related tools to support compliance assistance. This approach does not handle information about component properties. However, we have taken the idea of getting a common metamodel for different standards.

One of the limitations to the proposals made by the project, ModelMe! is that it was specifically created for the IEC 61508 standard and it does not take other standards into consideration. Their work has inspired this work in the idea of the concepts that should be included to model the standards.

4.3.2 Component's properties modelling

In the automotive industry the domain specific language EAST- ADL [EAST-ADL V2.1.12] already mentioned in chapter 2, includes a dependability package which provides support for safety information organization according to ISO 26262. The focus of EAST-ADL dependability package is part 3 of ISO 26262 standard. However, it does not provide support for the evidence management and the safety case class element is too simple to include all kind of argumentations elements that do contain other approaches. It also does not trace evidence used to support the claims for the safety case to compliance requirement for the standard.

4.3.3 Assurance cases modelling

This work has used the idea proposed in [Stensrud et al. 2011] where patterns are related to certification objectives. Their main interest is to take advantage of the goal-based structure of the safety cases and integrate the prescriptive elements of the standard so as to improve the transparency and consistency of the safety certification. The method used by Stensrud in [Stensrud et al. 2011] to transform the certification objectives into goals has been used to support the generation of prescriptive argumentation in the assurance cases presented in this chapter.

As mentioned in chapter 2, in the SACM standard "suppliers must not only ensure their delivery of adequate systems, but acquirers and users require the explicit, valid, well-

reasoned, and evidence-supported grounds for their confidence and decision making including related engineering conclusions and their uncertainty". SACM standard aims to provide "a framework for analysing and communicating the assurance arguments and evidence that relates to a system under consideration. Suppliers and customers can see how the system lifecycle products (system requirements, design, testing, field experience, etc.) relate to and satisfy the assurance requirements, enabling sufficient confidence to be gained in the behaviour and integration of the system within its operational context". However, SACM metamodel provides a poor support for arguments decomposition into modules and integration into a safety case architecture. We have used the SACM model to model arguments, but extending to improve its poor support to composition. The extension is described in detail in this chapter.

4.4 Components compliance process perspective

As we have previously mentioned components assurance modelling shall treat processes, artefacts and properties and how a components works up to a specific standard or a set of standards and guidelines. We have based our vision on extending the Common Certification Language (CCL) proposed by the OPENCOSS project [OPENCOSS D4.4]. The CCL defines a common conceptual and notational framework for specifying certification assets, as a means to get mutual recognition agreement and to be employed to discuss abstract notions from different domains. Using a common conceptual framework for different certification standards enables management of claims, evidence and arguments in a common format, sharing patterns of certification assessment, and allowing cost-effective re-certification between different standards. The concept of reuse of assurance assets relies on a clear understanding of three inter-related aspects of assurance: compliance management, safety argumentation and evidence characterization. The compliance management takes care of the level of compliance with standards and guidelines a component have work (such as DAL B, ASIL C or Class I). The evidence characterization handles the artefacts that a component should create during the development lifecycle and the safety argumentation makes a proper justification about the properties of a component. These concepts are further clarified below:

- **Compliance Management** encompasses the process and activities which are required to be carried out in order to demonstrate compliance to a standard/best practice/company standards by means of a series of defined deliverables specified in, for example, a compliance matrix.
- **Safety Argumentation** is the process by which the safety of a system is explicitly justified against a series of safety goals.
- **Evidence and Process Management**: both approaches depend on evidence - assurance artefacts as a result of an activity which support the claims made in the safety argument, or which can be used to fill out a compliance requirements objectives.

For this purpose the CCL was created. The goals CCL should achieve are:

- Getting mutual understanding of fundamental concepts of safety assurance and certification
- Reconciling argument-based and standards-based approaches to certification
- Devising, based on the common concepts, domain-specific “solutions”
- Facilitating reuse of safety assurance and certification assets

In Fig. 31 the CCL and its abstraction layers used are described as the structure to detail the concepts. In the “Conceptual Layer”, the meta-models for the CCL are defined. These models address the three main project concerns– compliance management, safety argumentation and evidence characterisation.

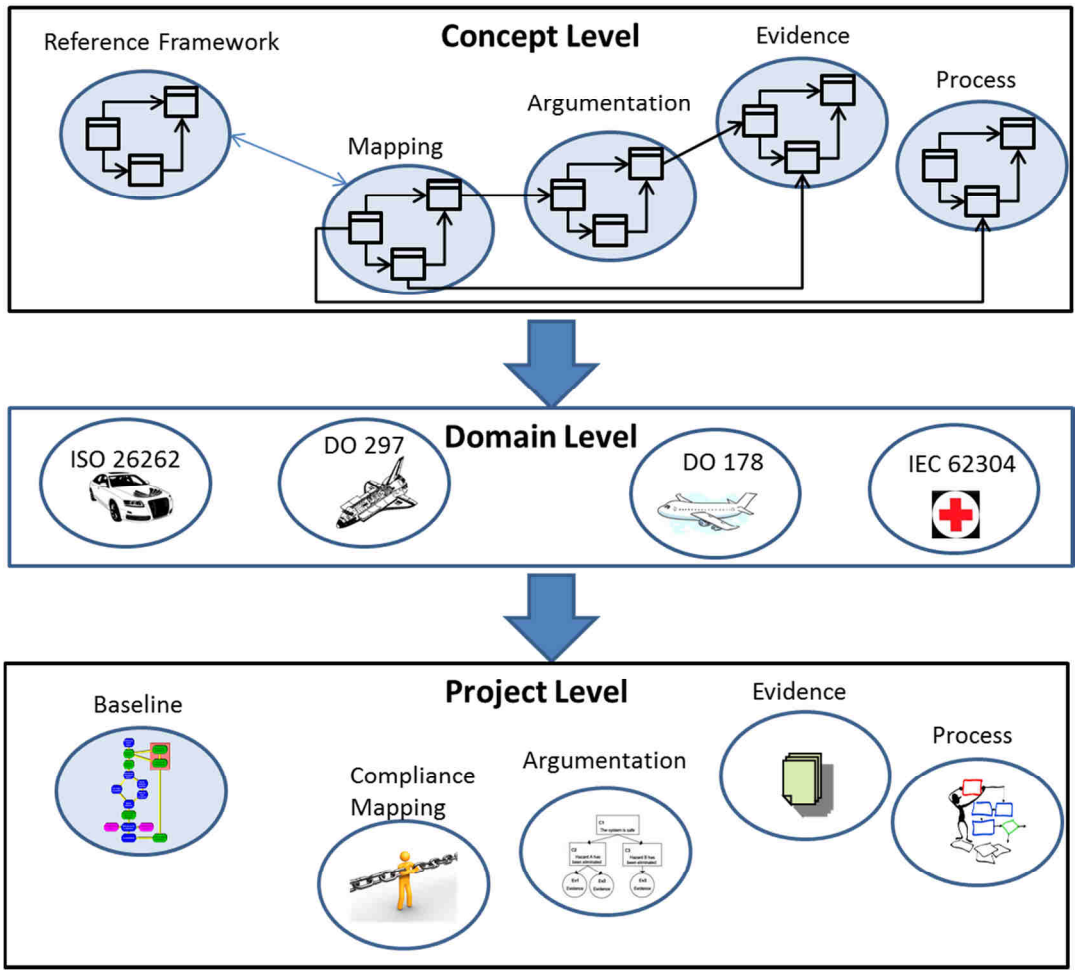


Fig. 31 CCL approach (3 layered structure)

In the “Domain Level” layer, standard-specific and domain-specific models are identified. These are models of the relevant assurance aspects of the applicable standards and guidance from the target domains, together with company and industry ‘best practice’ standards, to which assurance projects are developed and with which they need to demonstrate compliance. These models are developed in such a way as to conform to the

metamodels defined at the conceptual level, and will be expressed using terminology from the CCL.

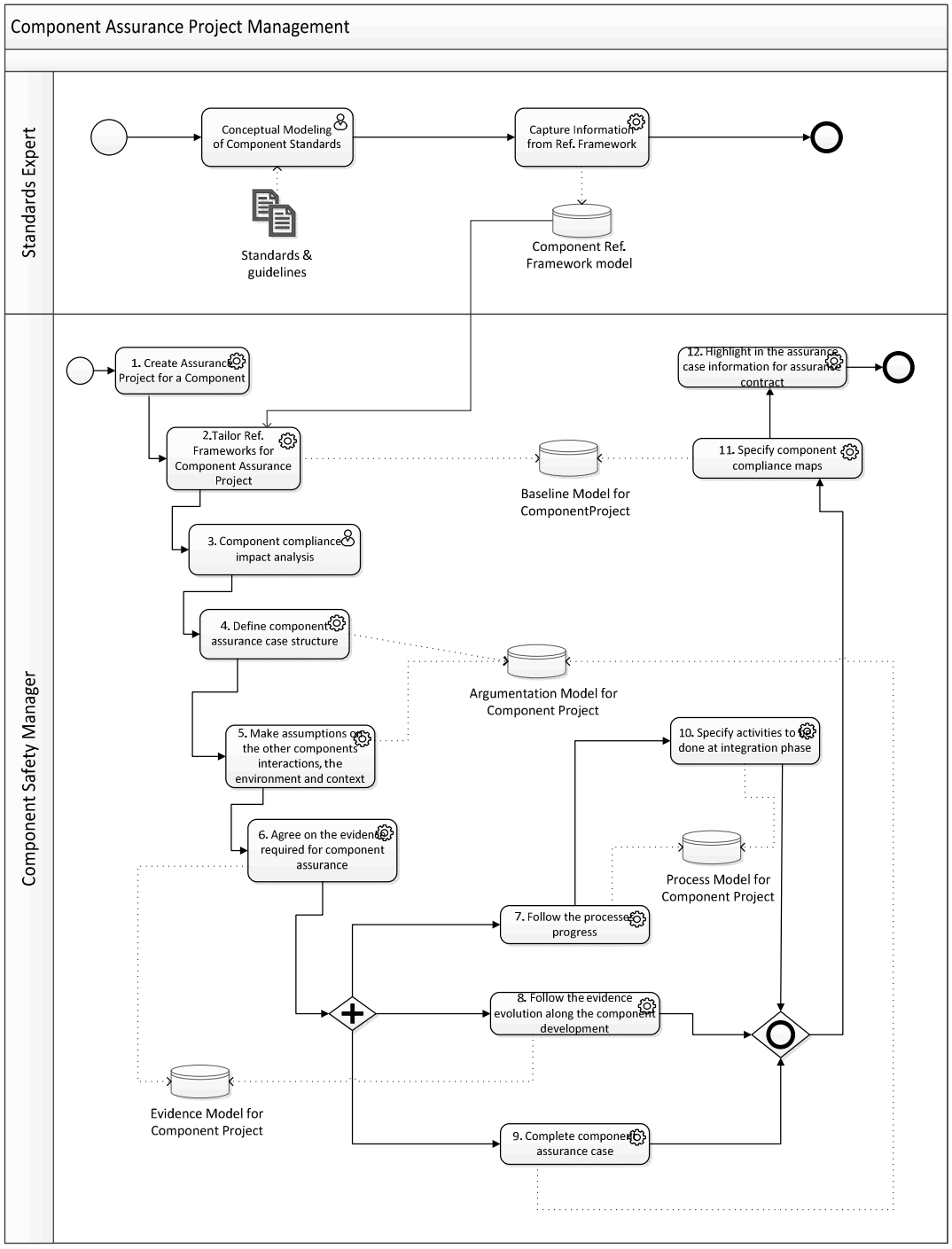


Fig. 32 Component compliance process

At “Project-specific” level, the project-specific process, assets and assurance artefacts, which are the actual target of the reuse effort, are captured. Metadata concerning these artefacts and the processes used to create them should be stored. This metadata is stored according to the metamodels specified at the conceptual level via the domain-specific models.

As we mentioned previously, in this chapter we are focusing on modelling assurance for the component development. A more detailed process for the component compliance development task is represented in Fig. 32.

4.4.1 Standard expert

The standard expert is the role in charge of the safety assurance and certification area concerned with the specification and handling of standards’ information, as well as any other information derived from or based on them (interpretations, tailoring, mapping between standards, etc.) This is done by means of Reference Frameworks. A Reference Framework represents criteria to which the lifecycle of a safety-critical system might have to show compliance.

Reference Framework specification deals with how to comply with a standard which is not the same as specifying the standard, it needs some further work and analysis on the standard itself. It is recommended that domain experts with experience on previous certification or assurance task take care of modelling the reference framework itself. The text of a standard does not exactly represent how to comply with it. For example on the avionics DO-178 standards it says “The objective of the software coding process is: (a) Source Code is developed from low –level requirements”. In practice a way to show compliance with this requirement is to trace from a specific requirement to the piece of source code implementing that specific requirement. In general, the text is structured in sections, clauses, tables, etc., whereas Reference Framework models the text and relates the concepts based on compliance requirements. For example, the standards usually present an overall lifecycle, but further reference activities might be identified in its clauses. The rationale why it is called reference framework and not standard framework is because sometimes we do not deal just with a standard but also with guidelines, circulars, company processes for standard compliance. In previous 3 Standards Analysis chapter it has been mentioned that for example in the case of an avionics application, the application should follow the ARP 4754 guidelines for safety Assurance assessment and the ARP 4761 guidelines for system concerning requirements. If the application is deployed in IMA architecture then the DO-297 standard should be followed with special attention to application related requirements. The application software should fulfil the DO-178 standard and in case it includes reusable software components, then they should follow the guidelines from the AC 20-148 advisory circular. If the application has specific hardware, that hardware should be DO-254 compliant. Thus for an avionics application, the compliance with two guidelines, three3 standards and one advisory circular is mandatory.

A detailed explanation of the Reference Framework meta-model is shown in [OPENCOSS D4.4]. Following we will describe with some detail some of the main concepts used to model the different standards and reference guidelines.

Reference Criticality Level: This concept references to the categories that indicate the relative level of risk reduction that needs to be provided for a reference assurance framework. Examples of this concept are: SIL (Safety Integrity Level), DAL (Development Assurance Level), and ASIL (Automotive Safety Integrity Level) levels correspond to criticality levels

Reference Applicability Level: This references categories of relevance or appropriateness that a reference assurance framework defines for its elements. Most safety standards define reference applicability levels in the form of recommendation (e.g., something is highly recommended), independence (between the participants of an assurance project), or control category (for the artefacts of an assurance project) levels. The specific reference applicability levels can vary among reference criticality levels.

Reference Activity: Unit of behaviour that a reference assurance framework defines for the system lifecycle and that must be executed to demonstrate compliance. Safety standards define different system lifecycle activities, from system inception to decommissioning, which correspond to reference activities.

Reference Role: This is a type of agent that participates in a reference activity. In AC 20-148 the activities and responsibilities from the developer of the reuse software component and from the integrator are described. Even if some standards (e.g., DO-178C) do not specify their reference roles, a company can define them.

Reference Technique: It is a specific way to create a reference artefact or execute a reference activity. For example, ISO 26262 specifies techniques that can or must be used during system lifecycle such as testing techniques for software validation. This information is usually provided in tables. Depending on the applied standard and the criticality level some techniques are mandatory whereas other for other criticality levels they are not requested.

Reference Artefact: Type of units of data that a reference assurance framework defines and that must be created and maintained during system lifecycle to demonstrate compliance. Data items, documentation, and work products correspond to reference artefacts for example for DO-178C, ISO 26262 …

Reference Artefact Relationship: Existence of a relationship between two reference artefacts. This is especially relevant for components; as this serves us to link system and components level artefacts. For example in ISO 26262 the item definition artefact includes the item definitions for the SEooC.

Reference Requirement: This is a condition or criterion that a reference assurance framework defines or prescribes to comply with it. The objectives specified in DO-178C tables correspond to reference requirements. In most of IEC 61508-based standards, the clauses specify reference requirements.

A Reference Framework mainly corresponds to the requirements to comply with, the process to execute, and the information to manage in an assurance project for compliance demonstration. People using reference assurance frameworks must be careful with the difference between the frameworks and the text of the standards.

We will model as an example the following excerpt of table A2 from DO-178c. The two rows represent the objectives related to the software requirements process. The first concept we model is "Software requirements process" as a Reference Activity. As mentioned, the objectives specified in DO-178C tables can be modelled as reference requirements, so next modelled concept is "High level requirements comply with system requirements" as a Reference Requirements. In Table 7 a column called "Output" is illustrated and indicates the data resulting for an activity. We will model "Software Requirements Data" as a Reference Artefact. Both the activity and the requirements should be fulfilled accordingly to a reference applicability level which is DAL C which we also model.

Table 7 Excerpt of table A-2 taken from DO-178C standard

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	High-level requirements are developed.	5.1.1.a	5.1.2.a 5.1.2.b 5.1.2.c 5.1.2.d 5.1.2.e 5.1.2.f 5.1.2.g 5.1.2.j 5.5.a	O	O	O	O	Software Requirements Data Trace Data	11.9 11.21	①	①	①	①
2	Derived high-level requirements are defined and provided to the system processes, including the system safety assessment process.	5.1.1.b	5.1.2.h 5.1.2.i	O	O	O	O	Software Requirements Data	11.9	①	①	①	①

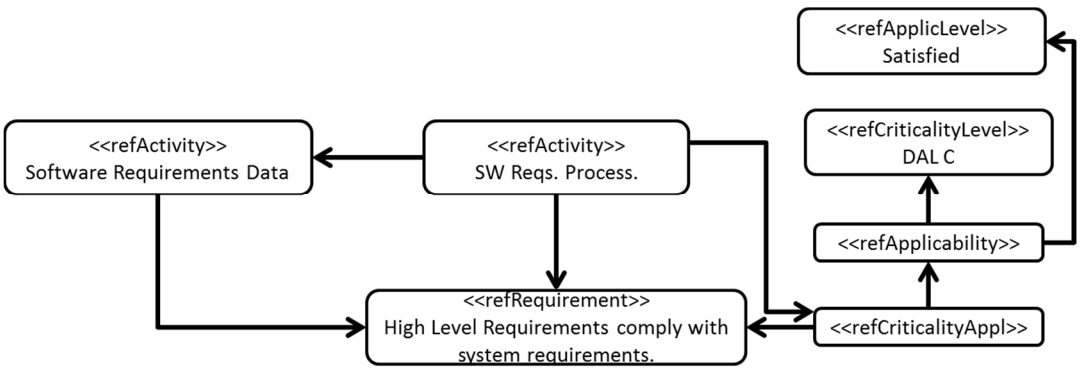


Fig. 33 Excerpt of a DO-178c reference framework

One of the first challenges we came into when modelling the different standards is a unique way, is the differences on terminology. We must take into account that the concepts (i.e., the name of the elements) of a reference framework correspond to abstract, generic notions of how to comply with safety standards and there is no need to follow the same naming conventions as a standard for a specific domain. Such notions are common to different standards and different application domains. For example:

- ISO 26262 work products correspond to reference artefacts
- DO-178C objectives correspond to requirements

When we decompose the reference activities and reference artefacts we should take into account that both reference activities and reference artefacts can be specified with different granularity levels. For instance, a reference activity can correspond to a system lifecycle phase, such as the system design phase and be later decomposed into other reference activities corresponding to sub-activities or tasks. In case of the reference artefact, one can correspond to a document and be later decomposed into document constituents. For example, a software test specification can be decomposed into test cases.

Table 8 includes a comparison of concepts modelling across different standard and the differences on terminology can be seen. However, there are points for comparison.

Table 8 Examples of reference framework metamodel concepts for specific safety standards

Concept	Standard			
	DO-178C	EN 50128	IEC 61508	ISO 26262
Ref. Criticality Level	Software Level A-E	SIL 0-4	SIL 1-4	ASIL A-D
Ref. Applicability Level	Objective satisfaction should be shown as Satisfied or Satisfied with independence, or is at applicant's discretion	Mandatory, Highly Recommended, Recommended, Not recommended, no recommendation for or against	Recommended, Highly , Not Recommended, no recommendation for or against	Recommended, Highly recommended, No recommendation for or against
Ref. Activity	Software development processes	Component design	Software design	Software unit design
Ref. Role	-	Designer		Designer
Ref. Technique	-	Modelling	Formal methods	Control flow monitoring.

Concept	Standard			
	DO-178C	EN 50128	IEC 61508	ISO 26262
Ref. Artefact	Software Requirements Data	Software Design Specification	System Design Specification	Software unit design specification
Ref. Artefact Relationship	Design Description *satisfies* Software Requirements Data	Software Component Design Specification *links to* Software Component Test Specification	Software System Design Specification *derived from* Software Architecture Design and Hardware Architecture Design Descriptions	Software Unit Design Specification *links to* Software Requirements and *specifies* Software Unit Implementation
Ref. Requirement	(11.3b) Independence: A description of the methods for establishing verification independence	(7.4.4.1) Software Component Design Specification for each component	(7.4.5.3) Software modularity, testability, and safe modification	3-8.4.5.1.1 Consistency and compliance of Functional Safety Requirements with respect to the safety goals

4.4.2 Component Safety Manager

The Component Safety Manager is responsible for Assurance Project Lifecycle Management which factorizes aspects such as the creation and maintenance of component assurance projects. Fig. 34 shows the compliance process the Safety Manager should follow to seek component safety assurance.

The first step is the creation of the component assurance project. The Assurance Project defines the assets produced during the development, assessment and justification of a safety-critical system, including those associated with justifying the safety of the system. This is done by the creation of a Project Baseline. A Project Baseline is a subset of reference framework (e.g., subset of a standard, part 6 software development of ISO 26262) that will be applied to a given assurance project, it indicates which standards and up to what level the component plan to comply with. An Assurance Project has three main elements: Baseline configuration, Permissions configuration and the Assurance Assets Package.

The Baseline Configuration has a set of Baseline Models. Each baseline model can reference to a specific Reference Framework model. In case of an avionics IMA software

component, the assurance project includes a baseline referring to the DO-178C standard for software and another baseline referring to DO-297. A Baseline model represents what is planned to do or to comply with, in a specific assurance project.

The Assurance Assets Package is a pointer to project-specific Artefacts models, and Argumentation models, and Process models. From the standard analysis done in chapter 3 Standards Analysis we have identified that the assurance information for a component should include data about activities, artefacts and properties. The Artefacts model includes information about the data created during the component development that will be used to show compliance. The process model includes information about the process followed for the component development. Finally, the argument model includes claims about properties of the component. These three models represent what has been done in a specific assurance project.

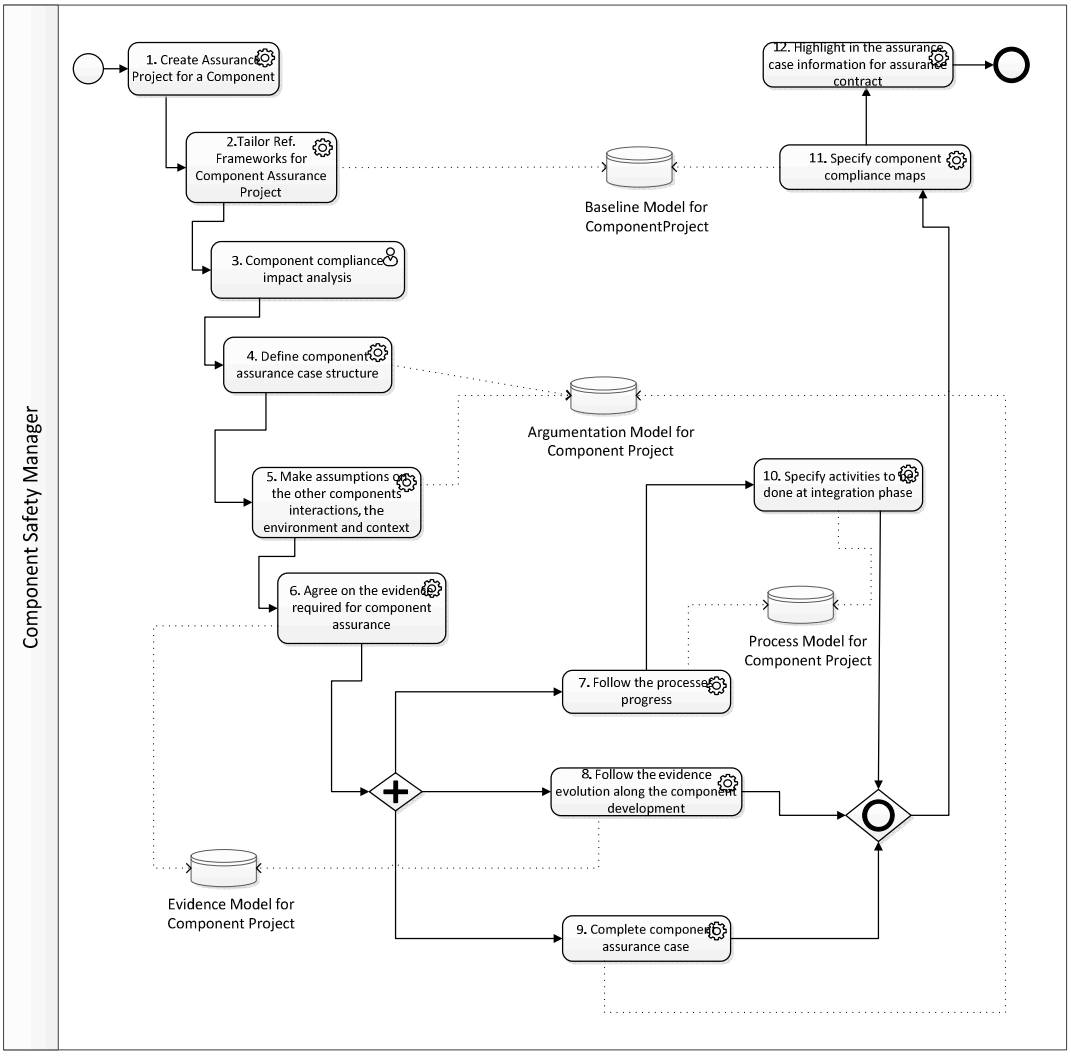


Fig. 34 Component compliance process

During step 2, the safety manager tailors the reference framework for component assurance. The baseline model gives us the capability to model and tailor the needs for compliance for a specific project. In case of a project for developing a software SEooC then not all the ISO 26262 standard applies and so the assurance project can be tailored to the specific needs of the project. It is similar to a project where we are developing an IMA application, not the whole DO-297 standard applies. In both cases, model-driven engineering is used along the project. This affects the means of compliance and in the assurance project and should be explicitly shown. In case it is decided that for a project model-driven engineering is used for a specific project, the verification activities can be tailored showing they refer to model checking techniques as compliance means.

There are not models to support step 3. In case the component is not a new development, the safety manager should perform an impact analysis and identify those activities, artefacts and properties that might be affected and modify the baseline.

During step 4, the safety manager should define the assurance case structure in the argumentation model. The assurance case structure defines the strategy followed for the component assurance case creation. We propose the use of the following argumentation pattern shown in Fig. 35 to structure the component assurance case.

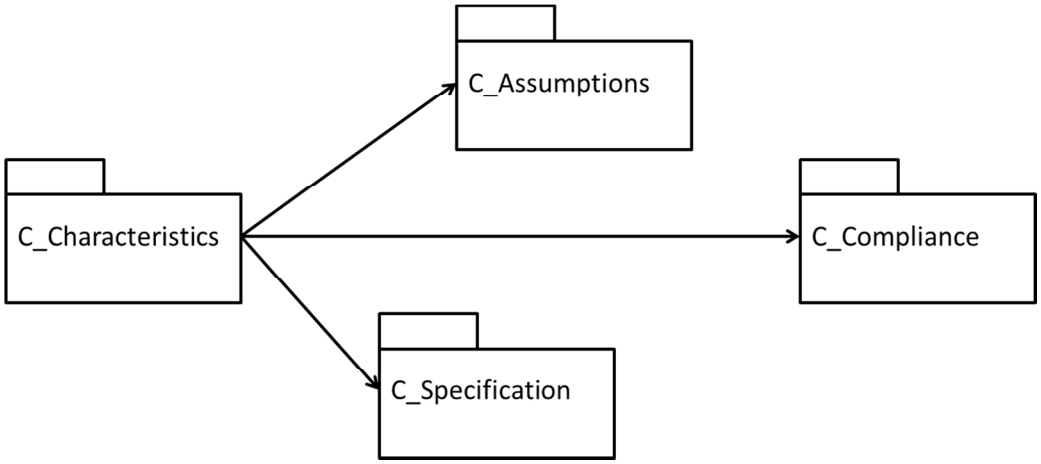


Fig. 35 Component Assurance case structure

The structure defines 4 argumentation modules:

- C_Characteristic: Includes claims about the component characteristics and properties and how they are derived or verified.
- C_Assumptions: Includes assumptions about the context and environment the component will work as well as dependencies for other components.
- C_Specification: Includes claims about the component specifications.
- C_Compliance: Contains claims about compliance means of the component.

The C_Compliance module is populated at this point. We propose an automatic argumentation generation based on the baseline decisions. The compliance means and objectives have already been specified in the baseline so the C_Compliance module just

have to refer to them. The generation is based on the transformations proposed in [Stensrud et al. 2013] and in [Denney Pai 2012]. In this way the reference activities model included on the baseline are transformed into top claims. The references requirements which those activities should fulfil are transform into sub-claims of those top claims. The sub-activities are also transformed into sub-claims and the artefacts are turned into information elements. The reference framework previously modelled as an example in Fig. 33, is transformed into the argument shown in Fig. 36.

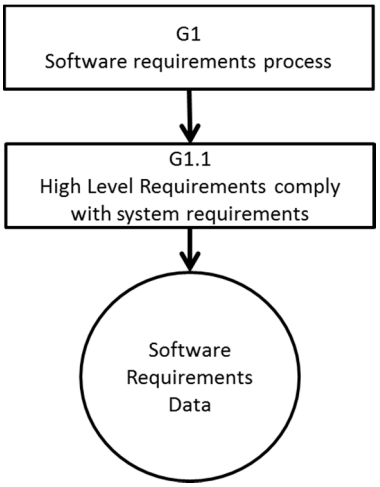


Fig. 36 Compliance argument generated after the transformation of an excerpt of DO-178c reference framework [Fig. 33]

Step 5 refers to the C_Assumptions argumentation module. The safety manager should include the assumptions made during the development of the component, the context and environment the component is designed to work in, interfaces with other components, expected intended use… The term "assumptions" captures a large and varied amount of information. There are design assumptions, from which the component specification was developed. In addition, there are usage and deployment assumptions, which need to be adhered to when the component is used (e.g. environmental conditions, input output values, failure behaviour of other components, expected hazards of the system...). We provide an assumption taxonomy that could support on this. In chapter 6 of this work we provide further details for this task.

In step 6 the safety manager should agree on the evidence the component will provide to show compliance. This is done by the support of the evidence model being this one the responsible of collecting and handling the body of safety evidence of an assurance project, including chains of evidence.

Safety evidence can be defined as the *artefacts* that contribute to developing confidence in the safe operation of a system and that are used to show the fulfilment of the criteria of a safety standard (e.g., safety analysis results, testing results, and source code). The evidence is used in the assurance project with two different aims: evidence used to support the compliance of an assurance project request and evidence used to support an

argument. Further details about the evidence model can be checked in [OPENCOSS D4.4], here we describe the main concepts of the model.

At this point the safety manager will specify the artefact definitions of the model. That is an abstract concept that represents an artefacts that's has not yet been created but will be used as evidence.

Steps 7, 8 and 9 are done in parallel during the component development. Step 7 follows the component processes progress is related to the activities to be executed during the component development lifecycle. The activities and the progress in those activities are modelled in the process model. The process management provides confidence on the results of the activity execution as it managed the process in a systematic way proving information regarding timing, people, techniques or relationship with other activities. Details about the process model can be checked in [OPENCOSS D4.4]

Step 8 is directly linked with step 6. During step 6 we have identified the artefacts that will be used as evidence while on step 8 we will handle the evolution of those artefacts during the component development lifecycle. At this time the artefacts model will be enriched with more concepts and relationships such as:

Artefact: The instantiation of the artefacts definitions previously defined. Concrete, individual, and identifiable unit of data managed in an assurance project. It represents an instance and version of an artefact.

Resource: The place, either electronic or not, where an artefact is stored. A hazard log can be maintained in a spreadsheet. Such a resource can correspond to an Excel file (format) in a local file (location).

Artefact Relationship: It explicitly specifies the relations among different artefacts. For example, the component requirement specification artefact has a relationship with the system requirements specification. Not only indicates if a component artefact is part of a system level artefact but also the relationship between the artefacts for the component. This is useful when doing an impact analysis as it can help the safety manager to detect possible artefacts affected by a modification.

Event: It represents a happening in the lifecycle of an artefact. Events are necessary to track artefact lifecycles. For example, the lifecycle of the final version of the hazard analysis and risk assessment report could be: creation, modification (e.g. change in the hazards analysis due to the inclusion of a new situation scenario), evaluation, modification (to address the result of the evaluation), evaluation, evaluation, modification (to address the result of the evaluations), and approval.

Evaluation: Specification of the result of making some judgement regarding an artefact. For example: The final hazard analysis and risk assessment report could be evaluated according to the following criteria: completeness, clarity, and reliability. As shown in the example for *Event*, some modification in the hazard analysis and risk assessment report might be necessary for addressing evaluation results.

The granularity of artefacts can vary: set of documents (e.g., system specifications), document (e.g., requirements specification), parts of a document (e.g., a given requirement), etc. The granularity depends on the purpose of an artefact.

When an artefact is used in an argumentation structure as evidence for a claim, such a use as evidence can have its own, emerging properties (e.g., confidence in the evidence).

Step 9 is related to step 4. The argumentation modules define the argumentation structure in step 4, while in step 9 we should complete and finalize the assurance case structure. In order to follow the best practices the safety manager can instantiate argumentation patterns. The argumentation model will be explained in detail in section 4.5 Argumentation Model - SACM extension for modular and pattern support.

Step 10 identifies the activities to be done at integration phase. The safety manager should specify the activities for compliance that are not done during the component development because they require that the component is integrated in the target environment to be executed. These activities should appear on the component project baseline model. They will be referenced again in chapter 5 Compositional Assurance Approach.

During Step 11, the safety manager should specify the component compliance maps. The compliance maps model is the mechanism to indicate up to what level a component complies with the baseline. The baseline specifies what it is plan to do and the compliance maps indicate the level of deviation from that planning. In the compliance map section we will describe this mechanism in detail.

Step 12 highlights the assurance case information required for the assurance contract. This is the last step before the component is released.

4.5 Argumentation Model - SACM extension for modular and pattern support

According to ISO 26262 a safety case is defined as “argument that the safety requirements for an item are complete and satisfied by evidence compiled from work products of the safety activities during development.” In fact in the automotive domain it is clearly stated that a safety case should be released in order to fulfil the ISO 26262 standard, and in fact it has been one of the artefacts that needs continue refinement along the project.

As previously identified in section 2.3. Assurance cases, there is one standard which is focused on the assurance cases modelling by the OMG. When analysing the latest version of the SACM [SACM 1.1] one of the lacks detected on this metamodels is that neither the argumentation patterns nor the modular argumentation are being supported. This could easily be solved by extending this metamodel, which is what we have done.

This section presents an adaptation of the OMG SACM meta-model, allowing us to present arguments in a graphical form similar to GSN, but using the richer concepts provided by SACM.

The first point to note is the modifications made on the SACM argumentation metamodel in order to include concepts for both modular argumentations and patterns. On the one hand, the modifications proposed here try to minimize the impact the actual SACM metamodel and, on the other hand, to include the concepts of modular GSN. Some modifications have been made also to facilitate the task of implementing the metamodel.

One of the first changes we have made is the inclusion of the agreement concept on the administration class container. As we know the Assurance cases are composed of arguments and evidences which support those arguments. This extension includes the new concept of agreement which is similar to what a contract should be.

The changes made in order to fulfil needs for modular argumentation and patterns are highlighted in green while the changes made in order to make it connect with other parts of the CCL metamodels are highlighted in blue.

Agreements are made between argumentation concepts. This is done to support arguments modularity. The mechanism to integrate the arguments modules, which in the end are argumentation concepts, is the application of the agreement. Agreements put in connection elements of the different argumentation to make a coherent and consistent argumentation.

In order to include modular argumentation, one element has been included, Agreement class. The Agreement class is a specialization of an Assurance Case element. The Agreement element represents agreements between parts (Argumentation).
Agreements are done between two or more Argumentation parts. It includes the premises and promises validated when both Argumentation are integrated. The argumentation class can be seen as an argument module.

The InformationElementCitation Class enables the citation of a source that relates to the structured argument. The information element citation can be used in the following situations: when we want to reference to an information element already supporting a claim in another argumentation but can also be useful to support a claim in the actual assurance case. However, this concept in SACM does not explicitly say what type of information is citing, that is why we have extended the concept including a type property.

The ArgumentElementCitationt Class cites an Argumentation, or an ArgumentElement within another Argumentation, for use within the current Argumentation. This concept is the key to understand the modular argumentation. There are times when it becomes necessary to be able to make a reference from the argument of one case module to some defined context that exists within the boundary of another, or to a Claim that is supported within another argumentation structure. Similarly to the InformationElementCitation Class, the ArgumentElementCitationt Class in SACM does not explicitly say what type of information is citing, that is why we have extended the concept including a type property.

The Claim clause is used to record the propositions of any structured Argumentation. This class has been extended from SACM; original definition. SACM proposes the use of the assumed property to indicate it is an assumption. We have also included the property public to indicate that this claim can be reference by others. In the component assurance case, those claims which indicate their property public to be true are considered the component guarantees.

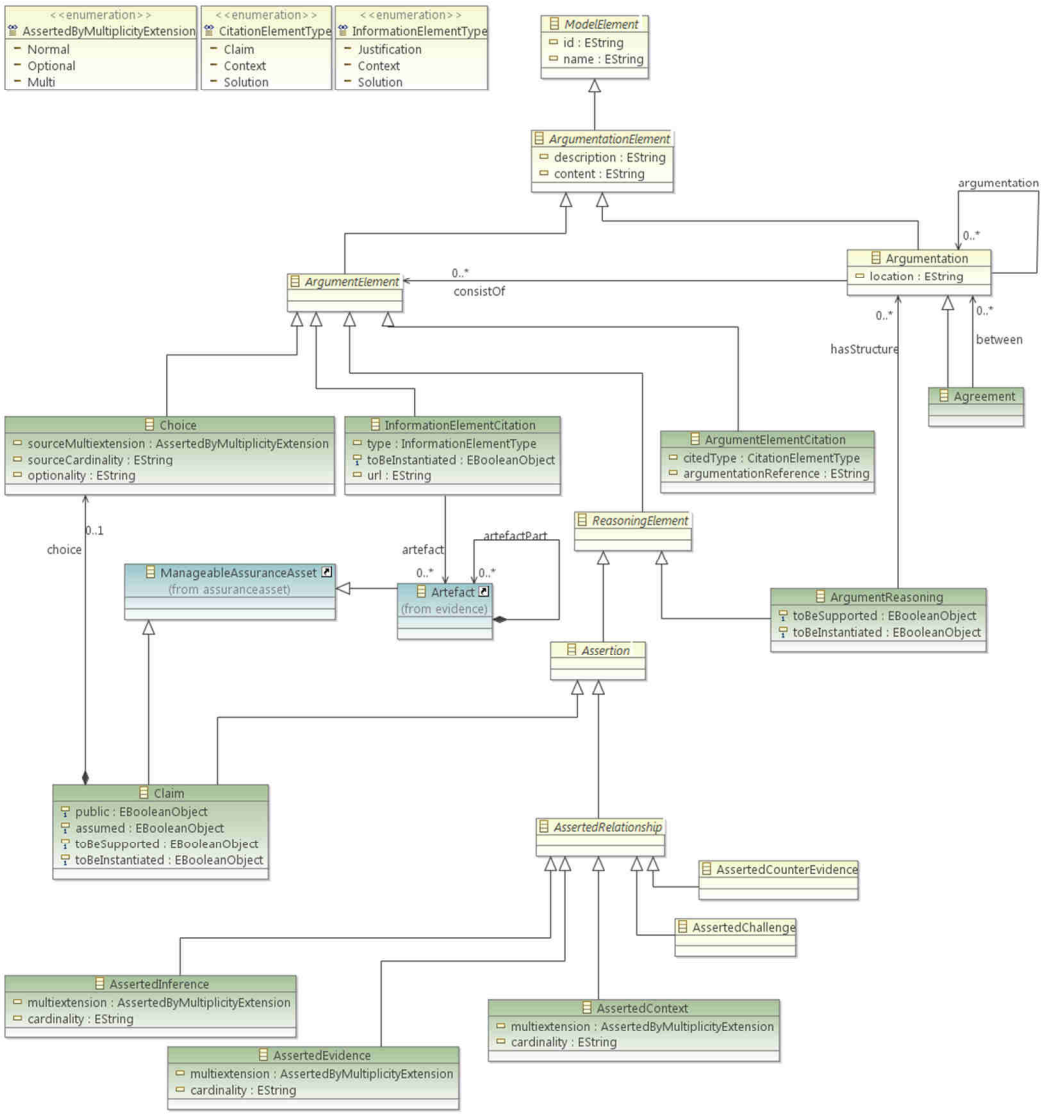


Fig. 37 Extension of SACM Argumentation Class meta model

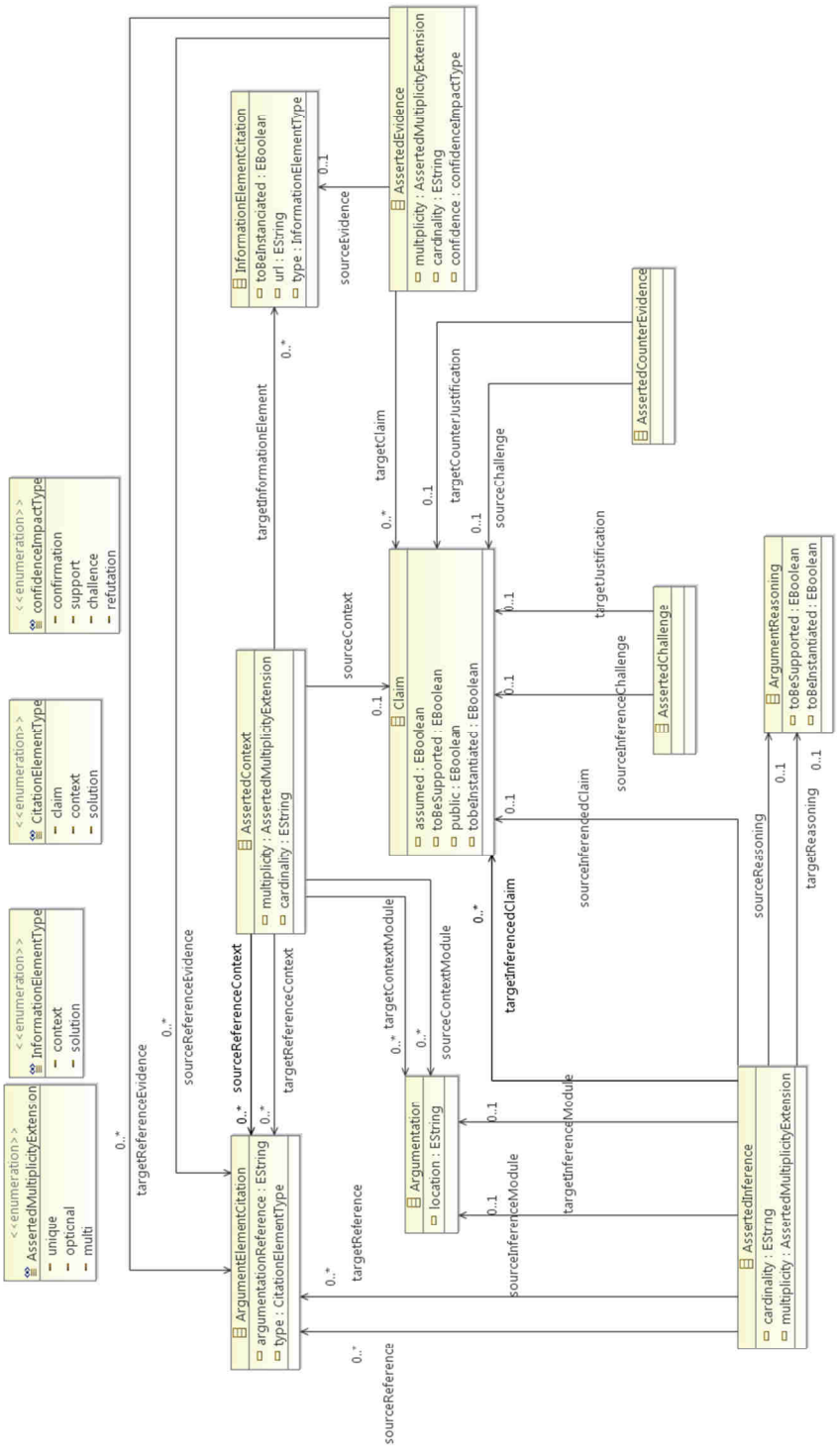


Fig. 38 Relationships view diagram

The following classes have also been extended: Claim, Argument Reasoning and InformationElement citation to include patterns related information. When they are intentionally declared as requiring further evidence or argumentation can be denoted by setting toBeSupported property to be true. They can intentionally be declared as requiring further adaptation and modification to be used in a new context by setting toBeInstantiated property to be true.

The Choice class which is a subtype of the AssertedInference Class has been added. It is used while developing argument patterns. It is used to denote possible alternatives in satisfying an inference. It is used to denote possible alternatives in satisfying an inference. It can represent 1-of-n and m-of-n selection, an annotation indicating the nature of the choice to be made.

The Asserted Inference, Asserted Evidence and Asserted Context association classes have been extended to include new properties so as to be used in patters. The multi-extension property indicates if the association is normal, optional or a multiple association. In case of a multiple association the cardinality property indicates the number of associations 1 to N.

The Asserted Challenge and the Asserted Counter Evidence classes do not have a graphical notation in GSN and SACM do not propose any. We have proposed a graphical notation for both of them.

4.6 Compliance maps

The compliance maps are part of the mapping model proposed in [OPENCOSS D4.4]. We identify a compliance map as a mechanism to indicate how exactly we are complying on a specific project to the reference requirements. Essentially, there are three types of mapping:

- Full Map – the elements in the mapping are identical.
- Partial Map – there is some similarity between the elements, but they are not identical (and there may be significant differences, depending on the context and requirements). In this case, a clear record of the similarities and differences is necessary, some quantitative indication of the "degree of map" should be provided in a form of justification. This is the case in which we need an artefact that should conform to a Standard "A" but as it is being reused from another project it is not completely mapping. We should identify to which level is compliant and what else need to be done to complete the mapping.
- No map – there is insufficient similarity between the elements to permit a mapping to take place.

We can make compliance maps for: activities, artefacts, requirements, roles and techniques.

Activity compliance maps indicate the activities in the process model that have been executed and comply with reference activities in the baseline. The level of confidence in

the compliance with the reference activities is defined with the type of map: no map, partial or full.

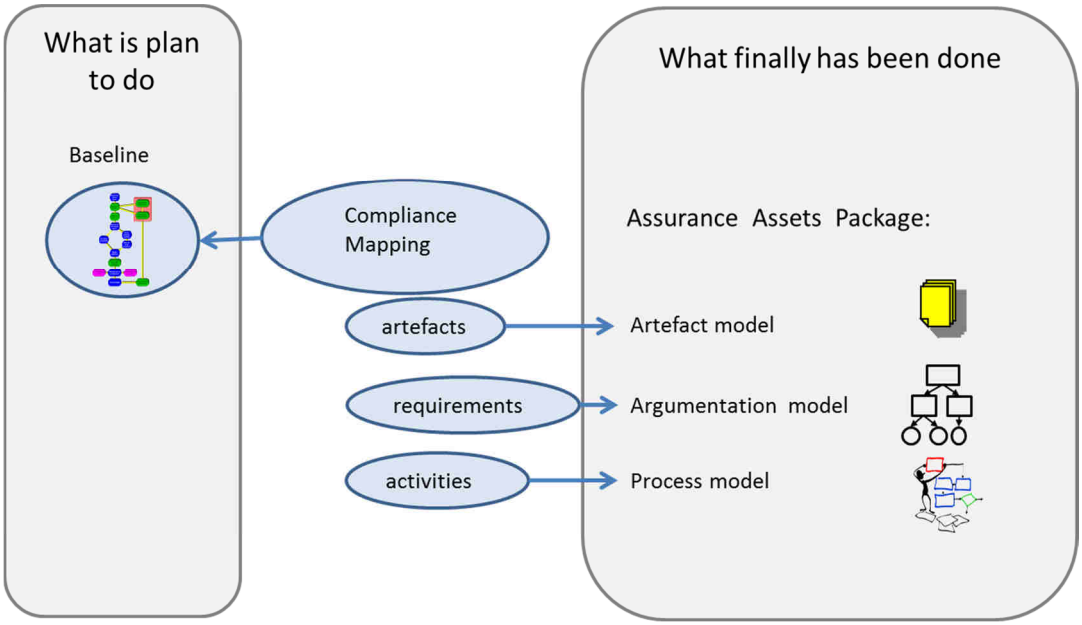


Fig. 39 Compliance maps

Similarly the artefacts in the artefact model are mapped with the reference artefact in baseline. The artefact definition is not valid for the mapping, as it is an abstract concept.

4.7 Tool support

We have implemented a tool to support the tasks for assurance modelling. The tool architecture and functionality is presented here [Ruiz et al. 2015_1]. Here we describe how the tool is used to support the tasks to be executed by the standard expert ad by the component safety manager. The complete user manual for the tool is available in [OPENCOSS D2.3]

4.7.1 Tooling for the standards expert

The first thing the standards expert should do is to create a Reframework Diagram following the wizard.

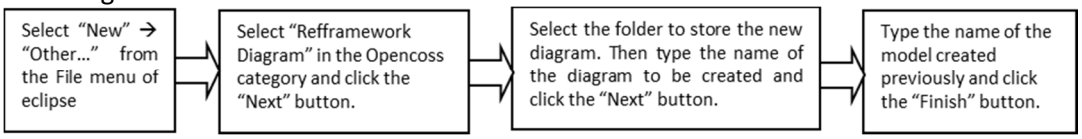


Fig. 40 Process to create a reference framework

After completing the Reframework Diagram creation wizard, the perspective of the tool will be opened composed by five views:

1. The **Repository Explorer** shows the contents of the repository.
2. The **Outline** shows the elements of the model and its edition.

3. The **Diagram Editor** the graphical modeller of a subset of concepts of the Reference Framework.
4. The **Palette** is a toolbox with the concepts of the model and the connections between them to add to the diagram.
5. The **Properties** to edit the properties of the element of the model selected.

The reference framework can be edited graphically by selecting the concepts in the palette on the right hand side of the editor and then editing the properties. The concepts with a graphical notation are the reference activity, the reference artefact and the role. These concepts have already been described previously. The connections indicate relationships between concepts.

Preceding activity connection links two activities and indicates the order of execution between the two activities.

Produced artefact connection links a reference activity and a reference artefact which has been produced as a result of executing the activity.

Required artefacts connection links a reference activity and a reference artefact which is necessary prior to the execution of the activity.

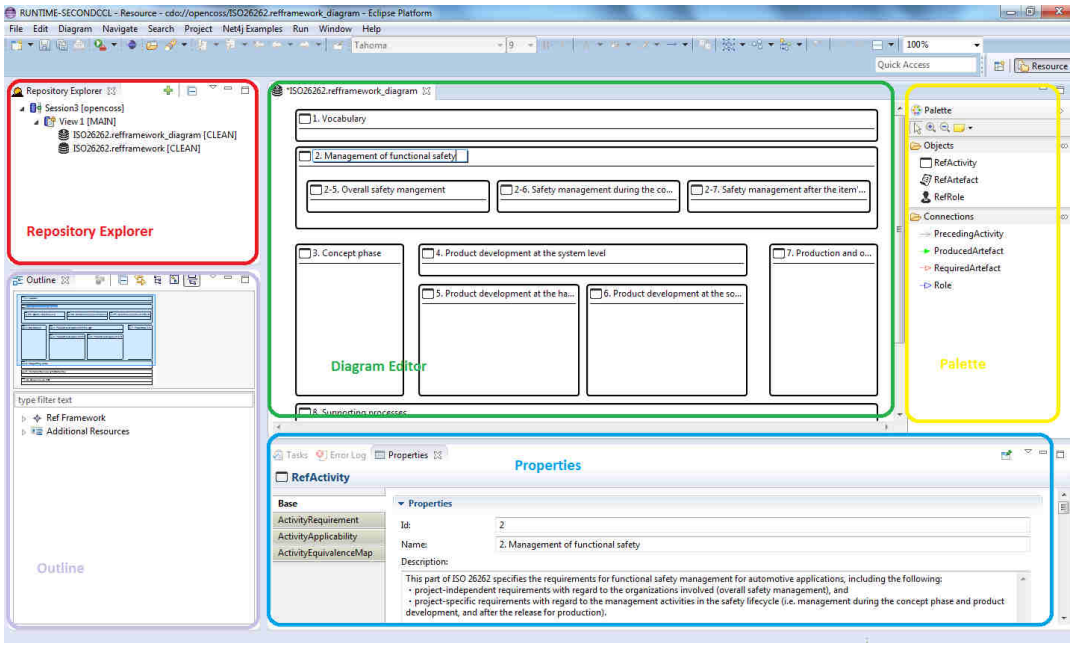


Fig. 41 Reference framework graphical editor perspective

Other concepts included on the reference framework model are accessible through the properties view.

There is also a tree view editor which can also be used for creating and modifying a reference framework.

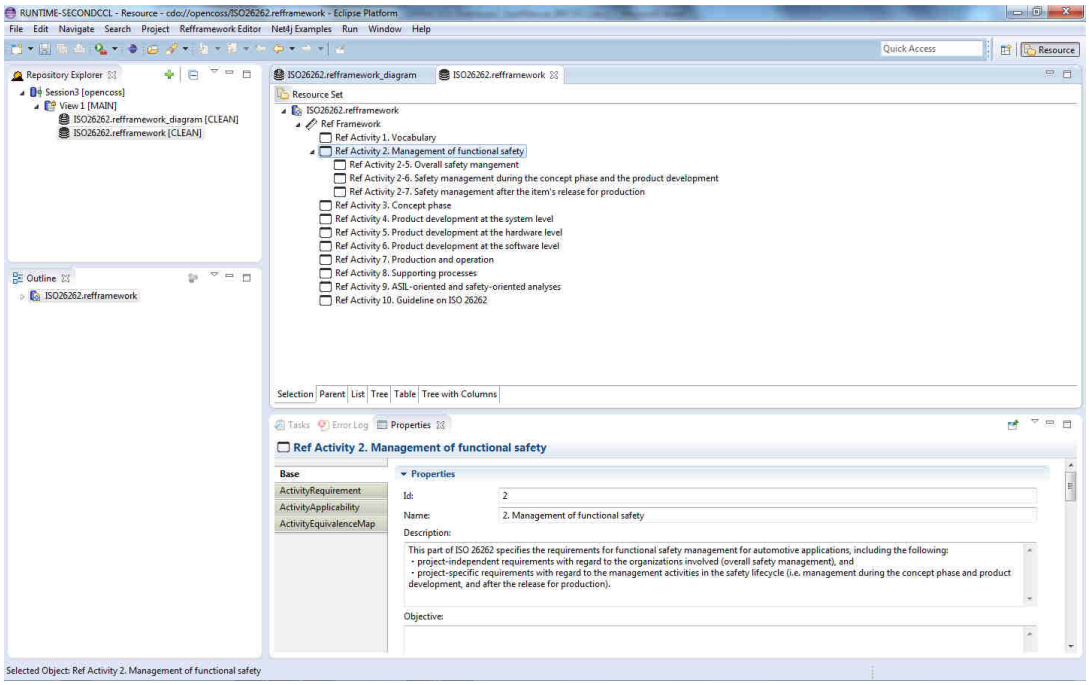


Fig. 42 Editing a Reference Framework in the tree view editor

The reference frameworks are stored in a database and can be retrieved and used when necessary.

4.7.2 Tooling for the component safety manager

Once we have a reference framework created, the safety manager can create a new component assurance project. To create a new assurance project go to the menu File→ New → Project and select New Assurance Project inside the OPENCOSS category. We will start the assurance project wizard.

The first page is to enter the name of Assurance project. The second page of the wizard will show in the left the list of reference framework models which are stored in the repository. Select the desired reference framework and in the right list will appear its contents in form of checkable tree for the generation of the baseline. Select the nodes of the tree that will be applied to the project are creating, give a name to the baseline and click the Finish button to generate all the project information.

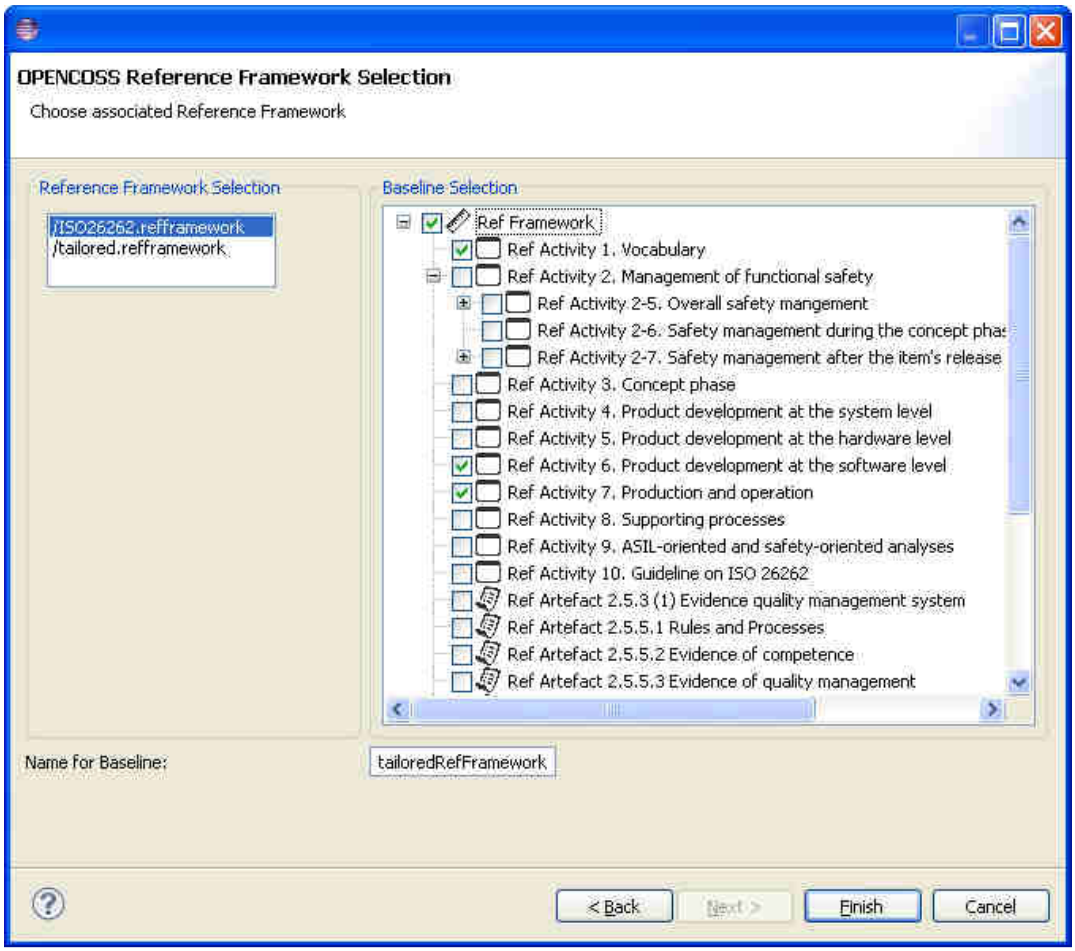


Fig. 43 Reference framework selection

Now in the Repository Explorer the new project will be displayed. The project is composed by 4 folders:

- Argumentation folder for storing the argumentation models with and argumentation model (with diagram) generated automatically based on the baseline's selected entities. As we mentioned on step 4 we are able to generate the compliance argumentation based on the selection made of the elements from reference framework at the assurance project creation time. This compliance argumentation can be modified and complete afterwards.
- Assurance Project folder that has the project information in .assuranceproject model, the baseline information in .baseline model(with diagram) and the .mapping model to store the compliance mapping information.
- Evidence folder for saving the evidence models
- Process folder for the processes execution.

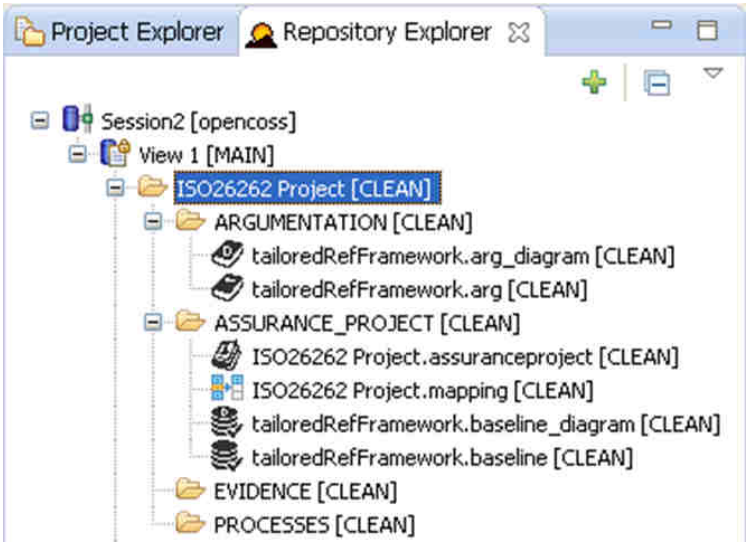


Fig. 44 Assurance Project structure

To edit the Assurance Project information double clicks over the model and its editor will appear. By default the assurance project has related all the models generated automatically, the baseline and mapping models in the active BaselineConfig and the argumentation model in the active AssetsPackage.

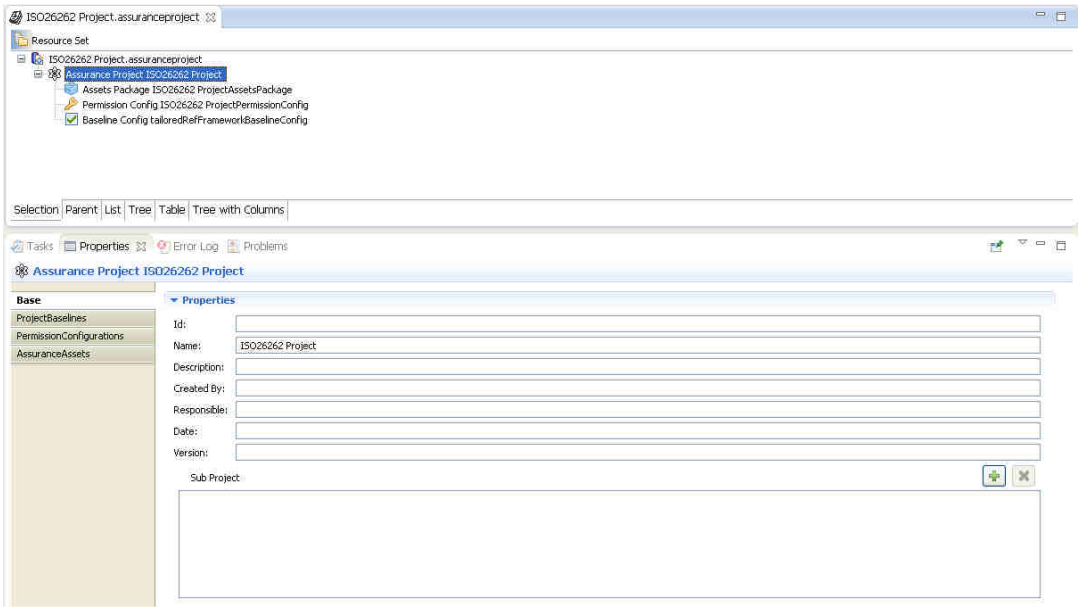


Fig. 45 Assurance Project editor

4.7.2.1 Baseline model edition

A baseline model is automatically generated when creating the assurance project. The editor is the same as the one used for reference framework creation.

To edit the baseline information in a tree view just double click over the .baseline model and its editor View will appear. The elements which have not been selected are displayed in the upper tree with a different icon, with a red cross on it.

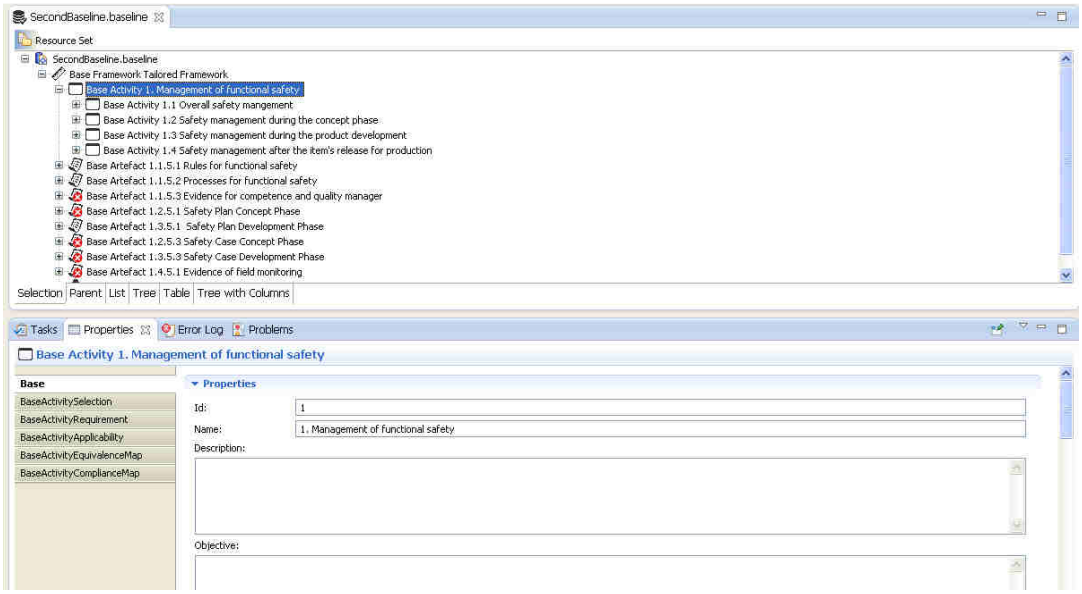


Fig. 46 Baseline editor

The baseline model can be edited also by means of a graphical editor, to use it double click in the .baseline_diagram model. The way of using this editor is exactly equal than the reference framework's editor.

The safety manager can create new baselines, for example if the component need to comply with more than one reference framework, the safety manager should create a baseline per reference framework.

To create a new baseline choose the wizard *Creates or Updates Baseline under the OPENCOSS category.* The first page of the wizard requests the selection of the assurance project model to update. The following steps are exactly the same than for the generation of a new assurance project. Select the desired reference framework model to be used as source for the generation of the baseline in the left list, then in the right list will appear its contents in form of checkable tree for the generation of the baseline. Select the nodes of the tree that will be applied to this baseline and give a name to the baseline taking into account that if the given name is the same as previous existing baseline, the contents of the previous one will be replaced with the information selected and the same will occur with the argumentation model. Finally, click the Finish button to generate the new baseline and argumentation models that will be added to the assurance project model and stored in the appropriate Assurance Project Folders.

4.7.2.2 Artefact model edition

Steps 6 and 8 are related to the evidence model edition. The first thing to do is to create a new evidence model. This is done by selecting *Evidence Model to repository under the*

OPENCOSS category. In the wizard select the folder where the evidence will be stored and include a name for the model. Once the Artefact Model has been created, the first item is presented to the user.

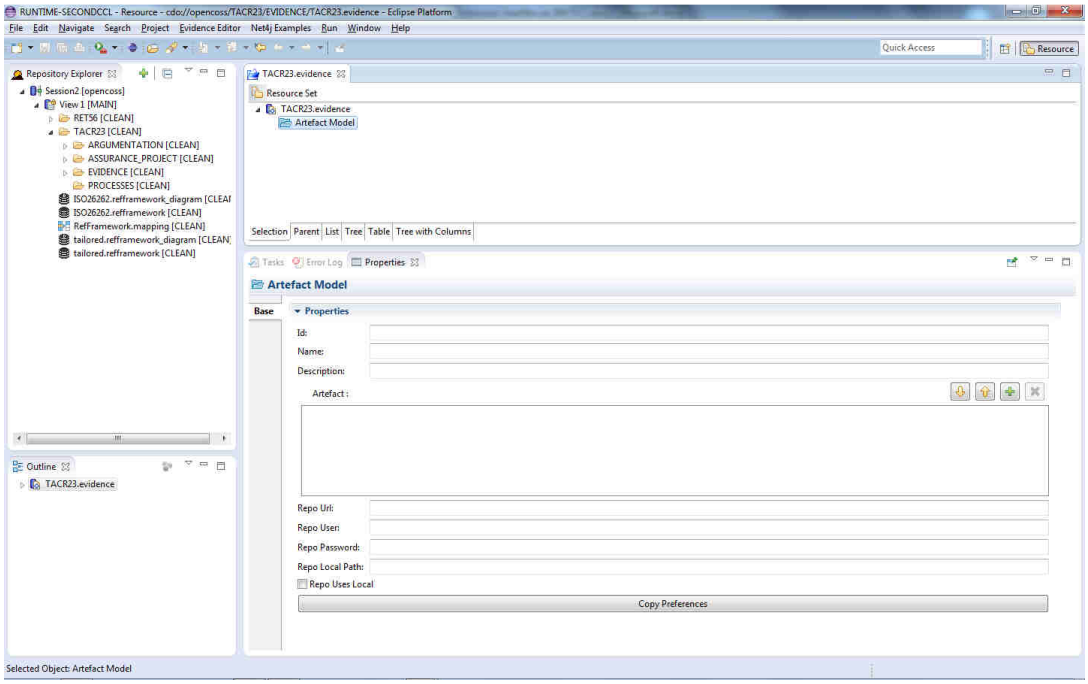


Fig. 47 Evidence model editor

For executing step 6, the safety manager should create the artefacts definition. In the artefact model editor, once the artefact model element is selected in the editor, in the properties view, in the Artefact field the component safety manager can add new artefacts definitions.

In the properties zone, the framework presents several fields to describe the new Artefact Definition divided in tabs.

For the step 8, the component safety manager can add artefacts to an artefact definition in two ways:

- Select the artefact definition, press the right button of the mouse and select the contextual menu *New Child –> Artefact*
- Or, select the artefact definition, select the Artefact Definition Artefact tab Properties, and press the button *Add*

When component safety manager modifies one Artefact, the system automatically adds to it an AssuranceAssetEvent of type Modification.

In the properties zone, the framework presents several fields to describe the new Artefact divided in tabs.

The component safety manager can also add a resource associated to an artefact in two ways:

- Once selected the artefact, press the right mouse button and select the contextual menu New Child -> Resource to bring up the Artefact File properties.
- Or, select the Artefact Version tab and press the + button.

In the recourse properties view the component safety manager can indicate the location of the file corresponding to an artefact.

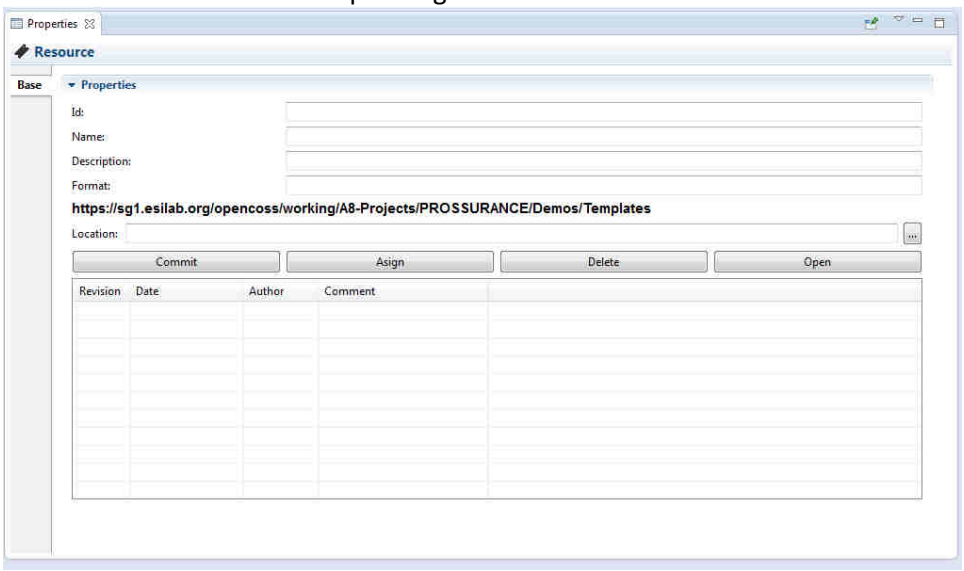


Fig. 48 Resource properties

4.7.2.3 Process model edition

For step 7 the component safety manager should create a process model by selecting ne process model under the OPENCOSS category and the wizard will start. It will ask about the name of the process model and selecting the parent folder to be stored.

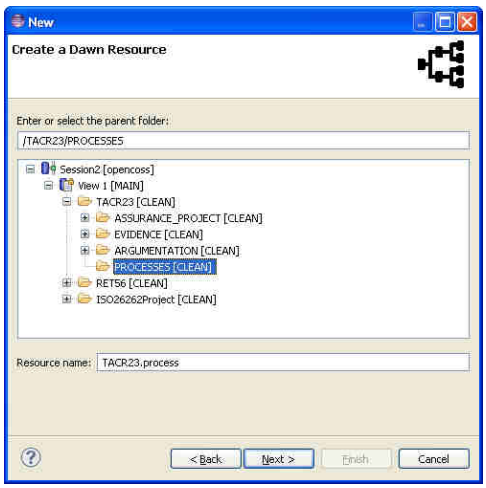


Fig. 49 Process model creation wizard.

Once the Process Model has been created, the first item is presented to the user. It is a similar editor as the one used for editing the artefact model.

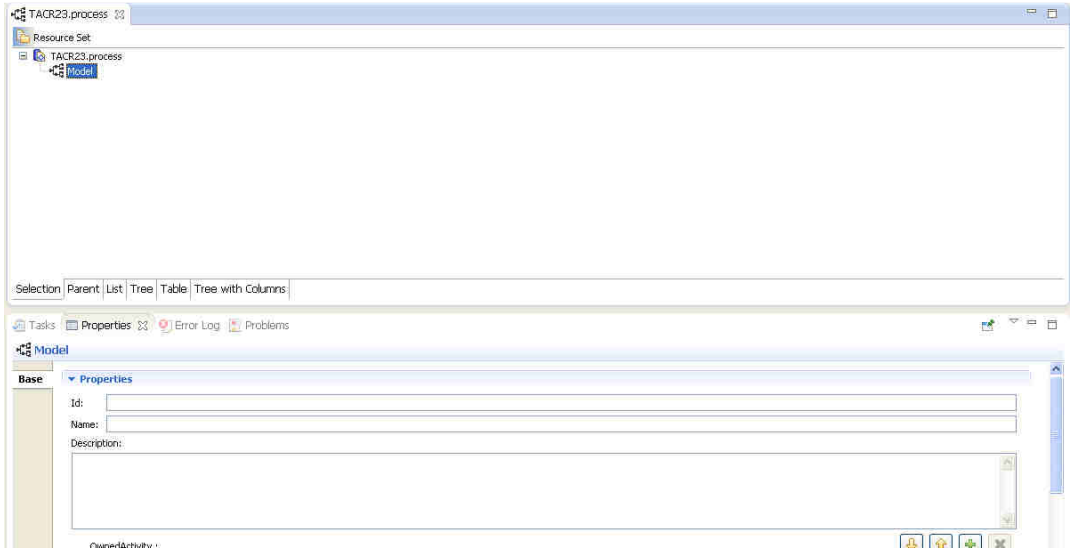


Fig. 50 Process model editor.

The Process Model allows defining activity, participant, person, tool, organization or technique objects. To create these objects, in the Model zone, click on the branch *Model* and press the right mouse button and select the contextual menu New Child or use its properties view:

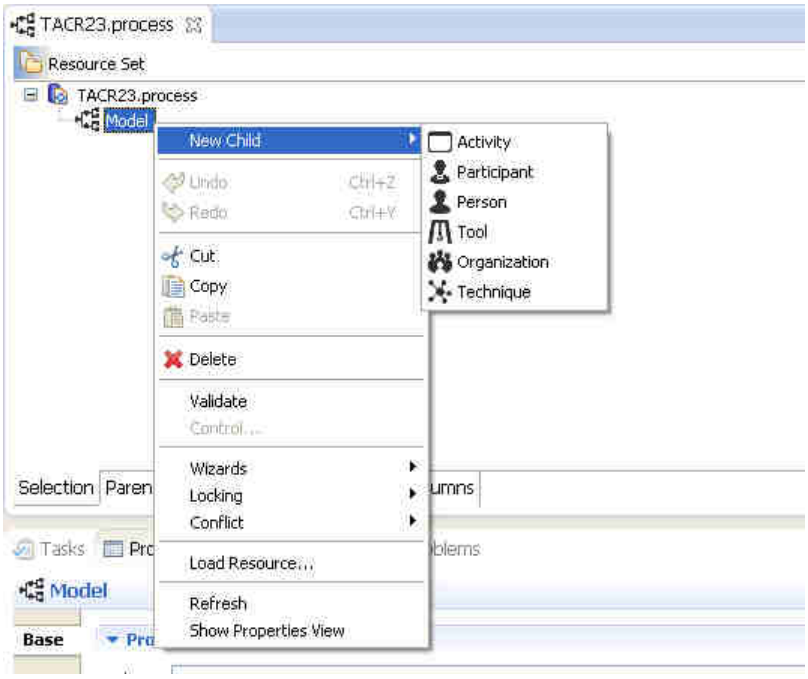


Fig. 51 Create Process Model data using context menu

4.7.2.4 Argumentation model edition

To create a new database-based argumentation diagram, follow the procedure below and generate a new diagram in the project folder.

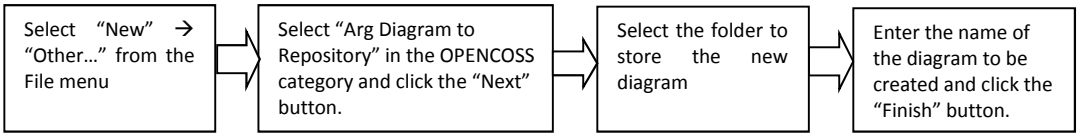


Fig. 52 Process to create an argumentation model

To open an argumentation diagram just double click on the Argumentation Diagram information file (.arg_diagram) to open a diagram in the editing window. The diagram can then be edited.

Nodes and relationships (or links) selected from Palette can be added to the canvas. Just select the node from the Palette, go to the editing window and select the place and size of the element.

The palette is structured into three different sections. Section "Argumentation core" includes the main nodes for argumentation. These nodes implement the GSN graphical notation. The "Argumentation relationships" includes all the different links between the different nodes. "Argumentation modular extensions" includes those nodes specific for the modular argumentation.

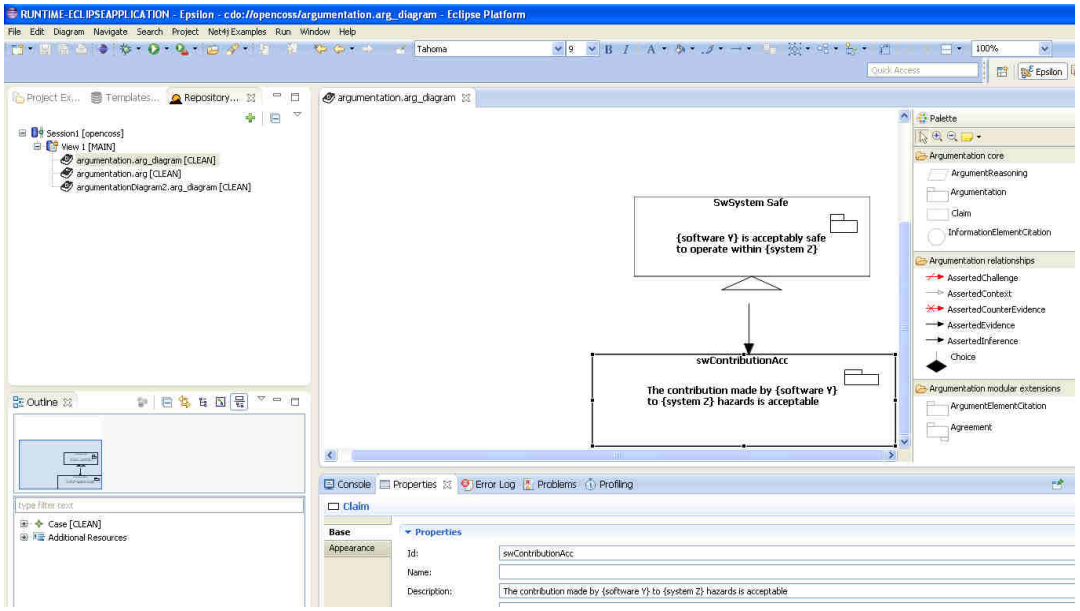


Fig. 53 Argumentation model editor

For step 9 the component safety manager can take advantage of the use of argumentation patterns. The list of available arguments patters is available in the Templates view.

When the user double click on an Argumentation Diagram file (.arg_diagram) will open the diagram in the editing window.

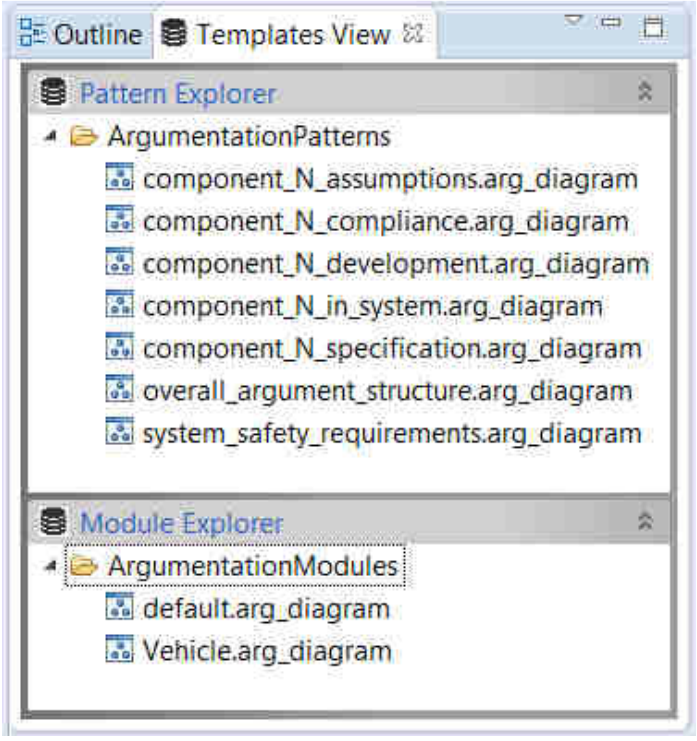


Fig. 54 Argumentation templates view

An Argument Pattern can be instantiated (thus all its content copied) into the diagram under edition. To proceed, drag a Pattern Diagram file from the templates view and drop it into the diagram under edition. Once you drop you will see the new elements that have been copied into your argumentation diagram.

4.7.2.5 Compliance maps edition

For step 11 the component safety manager should open the project baseline using the tree view editor. Then press the button “Mapping Set” on the properties form. This window automatically saves the mappings when checking or unchecking elements of the target baseline tree.

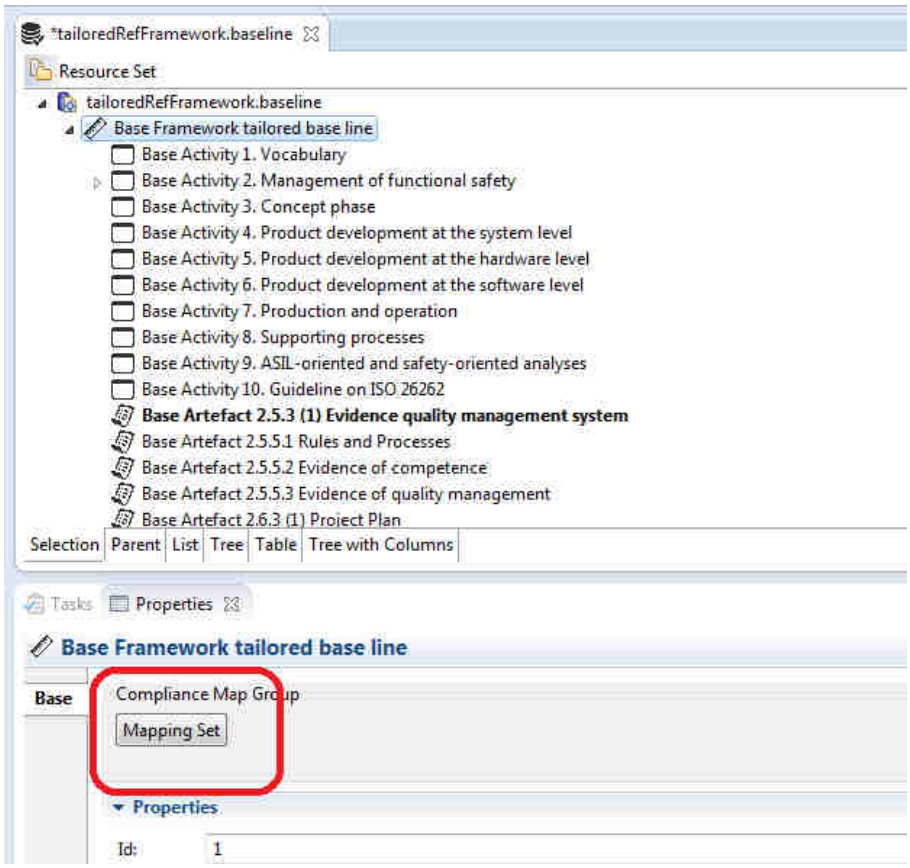


Fig. 55 Compliance map creation

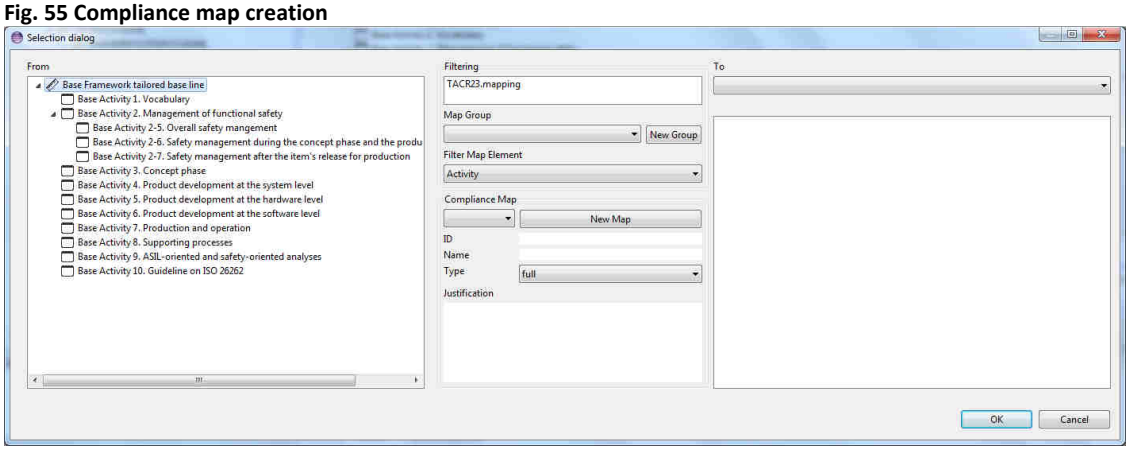


Fig. 56 Compliance map form

The Compliance Map form is organized in three zones:

- The *left zone* shows the actual baseline, and it loads the type of elements for which we want to make the compliance maps. For default, activities.
- The *middle zone* allows to make different filters like:

- *Filter Mapping Model* lists all the mapping models stored in the database, and it will be necessary to select one of them and one group model. It's also possible to create a new map group pressing the button "New group". This map group has to be part of the active Baseline Config of the project.
- *Filter Map Element*. It is possible to create compliance maps for activities, artefacts, requirements, roles and techniques, and the allowed maps are:
  - BaseArtefact -> Artefact
  - BaseRequirement -> Artefact , Claim or Activity
  - BaseActivity -> Activity
  - BaseRole -> Participant
  - BaseTechnique -> Technique

  When the filter changes, also the information showed by the reference framework changes. For example:
  - If the filter "Artefact" is selected:

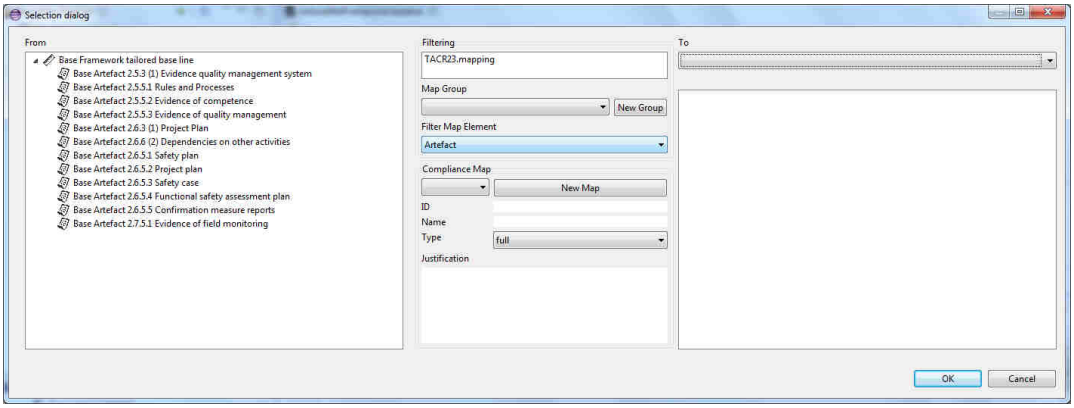


Fig. 57 Compliance Map, select map element

  Remember that these models have to be part of the active Assets Package of the project.
- *Filter Compliance Map*. This filter allows making different compliance maps for the same element.

- The *right zone* shows the list of models; depend on the map filter, stored in our database. We should select one of them. This selected model will be the target of the compliance map to create.

For making a compliance map, follow the next steps:

1. Select a mapping model and a map group.
2. Select the target reference framework.
3. Select the filter map element.
4. Select the element from the source reference framework.
5. Select o create the compliance map.
6. And for last, check or uncheck the element from the target model.

4.8 Conclusions

In Table 9 we have tried to summarize the issues and challenges which were identified at the beginning of the section and how we have tried to give them a response.

Table 9 Conclusions to challenges identified in chapter 4

Id	Challenge	Challenge Description	Mechanism	Solution description
1	Standard indication.	There is a need to indicate which standard a component should work to. Sometimes it is not the prescriptive parts of the standards which apply but advisory circular like the avionics domain where for reusable software component the AC 20-148 should be applied. This advisory circular is not prescriptive, however, it is hard to show compliance if it is not followed	Baseline	The baseline of an assurance projects reference to the specific reference framework- standard the projects aims to fulfil.
2	Level of compliance	We need to have the capability to indicate up to what level we are complying.	Mappings	Mapping traces the requirements on one specific project to the requirements specified on a standard. We can indicate if there is a full/partial or not compliance at all for a requirements Moreover argumentation also includes information about the rational for means of compliance of one requirement on a project to what it was required.

Id	Challenge	Challenge Description	Mechanism	Solution description
3	Evidence detail level	It is not clear to define up to which level of deepness these artefacts do goes and as a result of what activity or to support what claim have they been created.	Evidence model	The evidence model instantiation for a project will indicate the relationships of the different evidences
4	Different formats, not recognizable information	It is important from the industry point of view to provide the information required in a recognizable, standard format.	Reference framework	Modelling standards with a known model reduces the ambiguity and produces an harmonized way to show information among different stakeholders
5	Assumptions	The components is making assumptions about claims supported by other components and about evidences from other components that are used to support claims done at component level.	Argumentation	On the argumentation model, the concept of assumed claim indicates that you are making an assumption. Also the concept of information element citation indicated that that assertion will be supported by other component.
6	Guarantees	The components is making assumptions about claims supported by other components and about evidences from other components that are used to support claims done at component level.	Argumentation	On the argumentation model, the concept of public claims indicates that that assertion can be seen as a guarantee and reference by other components

"Any intelligent fool can make things bigger, more complex, and more violent. It takes a touch of genius — and a lot of courage — to move in the opposite direction." — *Albert Einstein*

5

Compositional Assurance Approach

5.1 Introduction

This chapter is focused on the composition assurance associated with integrating different components into a critical component-based system. Assurance is done at system level as it is a system property and as mentioned before it is hard to decompose across components. However, in the end the systems need to accomplish all assurance activities and show compliance with the safety standards, no matter if the systems is component-based or not. In this chapter we will try to find answer to the main problems that occurs when assuring a component-based system in a critical context according to functional safety standard.

In the previous chapter we have mainly talked about assurance at component level is managed. We have defined how assurance should be managed inside a component in order to get the assurance assets to be provided at system level. Reuse is taken into account from the development phase. In this chapter we move on a higher level as we try to describe how assurance should be done at system level managing all information which is taken from the different components that are part of the system.

This chapter shows the integrator point of view where a previously developed component using the assurance approach presented in the previous chapter is incorporated into the final system where it will work. We try to express the hierarchy of compliance responsibilities across the system components in a systematic and unique way.

Here, we show the application of the approach on simple examples extracted from industry situations so the concepts are better understood when put them into practice with tool support. These are not complete use cases but some excerpts of common situations, that industry usually faces and how the presented approach can be used on these situations.

This chapter is organized as follows. First, the challenges section introduces the main problems the industry and the integrator are facing. Following, there is a context literature review on the existing approaches regarding composition assurance from the integration point of view. We try to see how the problem has been treated for other aspects such as design or argumentation and how we can take advantages of those solutions on this new situation. Next section is about how standards from different

domains handle the integration of components. Then, we talk about the mechanism proposed to be able to model the requirements for composition and components integration taking into account concepts described on previous section. We define some rules we should follow in order to make composition in a systematic way. Then we described some examples where this can be applied on common industry situations. Finally we describe some conclusions and resume how we have fulfilled some of the issues described at the beginning of the section.

In chapter 4 Assurance Modelling we have described compliance processes for component-based systems from a high level perspective. In this chapter we focus on components compliance along the component development. In Fig. 58 the activities that are covered in this chapter are highlighted.

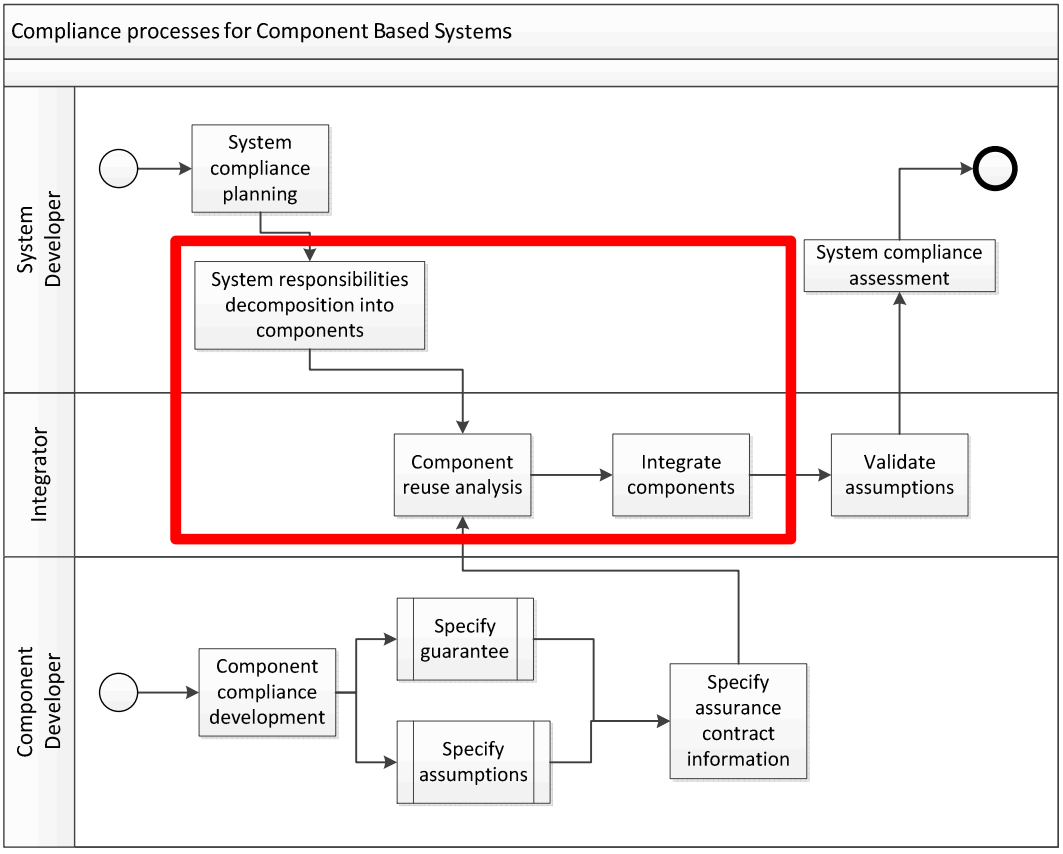


Fig. 58 Scope for chapter 5 in the compliance processes for component-based systems

This chapter focuses in the activities done by the integrator prior to the system validation and release.

5.2 Challenges

In the composition phase we deal with the reuse of a subsystem or component into a new context. When an already-assessed system has to be reused, even in a new context, then

some pieces of evidence might become inadequate or new evidence might have to be provided.

Innovation and productivity in safety-critical systems is curtailed by the lack of affordable (re)certification approaches. Major problems arise when changes to the system entail the reconstruction of the entire body of certification arguments and evidence.

In Fig. 59 it is describe how assurance is treated during the integration of a component in a new system.

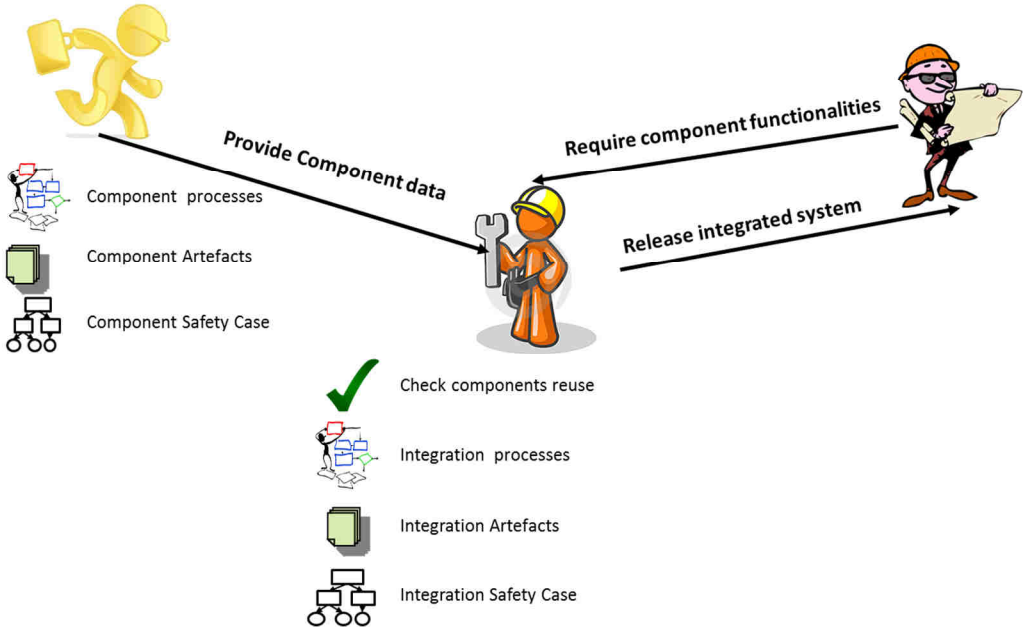


Fig. 59 Composition assurance scenarios

Responsibility decomposition

The theory of component-based systems is easily applied here. However, in critical systems we need to assure properties such as reliability, availability, safety, security and standard compliance. When decomposing the system into smaller parts usually developed by different teams and even different suppliers, responsibilities are hard to assign. Whole system assurance is not possible to be accomplished without the support of component assurance.

In [Espinoza et al. 2011] we mention that: “The challenge in such systems is to assess not only the certifiability of each component or module, but also its certifiability once it is in an ‘integrated’ state”. If the assurance case relies on justifying the properties of components, then we should ensure that the system has been built out of the specific component for which there is evidence that the component has the necessary properties. “The mean to transfer a common frame for functional and design characteristics of a component from provider to integrator for the compatibility/gap analysis would bring a big benefit to the embedded system community for sharing components and increasing

the safety by the broad service history." Espinoza identifies some of the main problems teams need to solve when reusing a component or a sub-system in a new system. It is important not only the component by itself but the system in which it will end up being used. In critical systems is the ultimate system what in the end succeeded on the certification process.

We start by reusing a component in a system which need to comply the same standard as the component original did on its development. The context and environment match with the ones assumed and verified on the first time the component was used. However, the first variation appears when the assumptions are not directly compatible and further analysis should be done. In this case, analysis on the reuse viability is necessary so as to check after a throughout analysis on the assumptions validation and impact.

Reusability on the same context but taken into account a new standard often happens while reusing a component. They need to be adequate to the system level standards.

Teams are focused on their subsystem development as their development will end up working on a fully integrated system. In order to make their development they need to make assumptions on the behaviour of other components.

Need to reference to other component activities

Components might be developed by different teams or suppliers, however, on their assurance processes they do need to reference other components or the system development activities. For example, the advisory circular AC 20-148 [AC 20-148] includes as information the reusable component developer should provide to the integrator: "… verification procedures, especially for verification activities that the integrator or applicant must repeat for the integrated software installed on the target computer environment". Component assurance will not be fully accomplished until the components are part of the whole system and the entire assurance activities takes place.

Reference to other components evidences

Not only does a component need to reference others team's activities but also evidences. Results from analysis would not have any sense without the context, or safety constraints defined and verified on others component. An example of this evidence reference appears in the automotive domain when developing a SEooC. The hazard analysis and risk assessment process done at concept phase is done at vehicle level. However, when developing a SEooC we should reference the analysis done at item level.

Evidence refinement by composing artefacts from different components

Evidences from a component are necessary to refine artefacts used as evidences at system level. In this sense safety analysis like a component fault tree analysis is required to complete and refine safety analysis at the whole system level. Plans at system level should reference the use of the reusable components and the evidences provided to integrators.

Need to show integration from the assurance perspective

It is important that component integration is done not only at technical level but also assurance should be addressed. Zimmer [Zimmer et al. 2014] defines interference as "a failure propagation scenario in which a failure of one software component propagates to another software component via the platform's shared computational resources." This is related with the concept of assuring freedom of interference, a property requested by functional safety standards.

Capability to trace what each team is doing regarding assurance

From the system responsible point of view it is important to see the progress of the whole project, the fully accomplished system, how each of the team is going. If we can consider the system as a set of components working all together, then the assurance can be seen as the addition of the assurance activities for each of the components. Of course this is a very simplified vision; however, it does highlight the need to trace every team work regarding assurance. This is not only required for responsibilities assignments but also for planning and monitoring work. Estimations on efforts and costs at the system level are impossible to define without these inputs.

5.3 Analysis on existing approaches

There is a special interest on modular certification as is aimed to be feasible. Initially this is done by informal reuse of safety arguments in relation with a function from one application to the next, and this is aligned to the way systems is actually developed as well—as separate components with interfaces between them [Rushby 2002].

Kelly in [Kelly 2001] proposed a modular and compositional construction for safety cases that can be adopted to create a modular safety case architecture for IMA-based systems to correspond as far as possible with the modular partitioning of the hardware and software of the current system. In the same way as system architecture is design, it is possible to establish interfaces between the modular elements of the safety justification such that safety case elements may be safely composed, removed and replaced. Similarly as with system architecture, it will be necessary to establish the safety argument infrastructure required in order to support modular reasoning. In this work we have inherit the idea proposed by Kelly by adopting a modular view aligned with the system architecture.

One of the main challenges that modular certification has in contrast with the modular design is that certifications must consider not only the regular operation but also abnormal operation and malfunction components [Rushby 2007]. Rushby affirms in [Rushby 2002] the problem is that conventional design and the notion of an interface, are concerned with normal operation, whereas much of the consideration that goes into certification concerns abnormal operation, and the malfunction of components. More particularly, it concerns the hazards that one component may pose to the larger system, and these may not respect the interfaces that define the boundaries between components in normal operation. In order to cope with this we propose the use of

assumptions and guarantees in the component assurance development that will be checked during the compositional phase.

5.4 Compositional Assurance

Component-Based Software Engineering (CBSE) is seen as a common and well known strategy for dealing with complex systems. As systems grow in complexity, so does the trend in using components. In CBSE approaches the concept of contract appears as a means of connecting components together thus allowing them to interoperate. More specifically, contracts record agreements in terms of assumption and promises as is described in [SPEEDS D2.5.4]. Assumptions are the functionality that a component requires from other components in the system, and promises are the functionality that the component can offer to other components in the system. A component's assumptions need to be validated before its contract promises can be fulfilled.

There are challenges when using this approach for dealing with safety, and safety assurance, properties. Safety is a system property and because of that, it can be hard to define the contribution of components that have an impact on safety. Contract specifications addressing safety have been proposed in the past regarding modular safety case development.

From the safety perspective, safety and assuring the correct function of components does not mean that the (composed, integrated) system will remain safe. The context in which the component is going to be integrated is important, and as we indicate in [Ruiz et al. 2013-1] for the SEooC (Safety Element out of Context) perspective, the assumptions of the item can be understood as the context characterization. In addition, to support safety assessment, failure behaviours of components, and their behaviour in the presence of failures, must be defined. 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 we also need to define the set of claims that need to be made concerning a component to support its certification against a particular safety assurance standard. Different standards address this problem in different ways. In ISO 26262 Development Interface Agreements (DIA) are described as a way to specify both procedures and responsibilities allocated to distributed developments for items and elements. The DIA includes information beyond technical safety by addressing procedural and confidence related issues. In the avionics domain we can find similar requirements while talking about modules and application reuse on an IMA (Integrated Modular Avionics) platform. In DO-297 [DO-297] for reuse of component acceptance it is required that component limitations, assumptions, etc. are documented and a usage domain analysis is performed to ensure that it is being reused in the same way as it was originally intended.

Guidelines and standards prescribe the information needed to manage at the assurance project level as we have indicated in chapter 3 Standards Analysis. When analysing guidelines and standards, we noticed that the data required for assurance can be classified in three main categories:

- *Artefacts*: referring to the data required by an entity when doing the safety assessment
- *Properties*: these are characteristics that must be present in the component and after the integration in order to confirm that there are no concerns or 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.

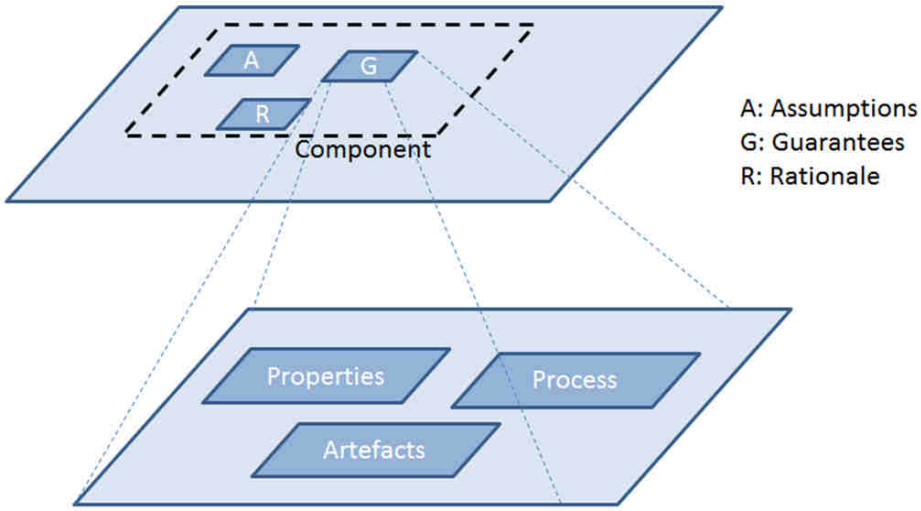


Fig. 60 Assurance contract view for a component

5.4.1 Components integration compliance process perspective

Fig. 61 shows the activities performed by the integration in the context of compliance processes for component-based systems in detail.

In the previous chapter we have agreed on the creation of an assurance project per component to develop. The Common Certification Language (CCL) relies on three aspects: Compliance management, safety argumentation and evidences management as described on previous chapter. Composition can take advantage of the CCL and relate those aspects regarding how assurance at the whole system level could be composed by the addition of all the information for each of the components integrated with the system.

The standard expert should also play a role in this process. The expert is the responsible to model the standards and guidelines for composition, the focus here is the integration of the different components and the division of responsibilities between the component safety manager and the integrator. When we talk about component composition, we highlight relationships that exist between concepts from different reference frameworks. These frameworks are related to each of the components that are being integrated.

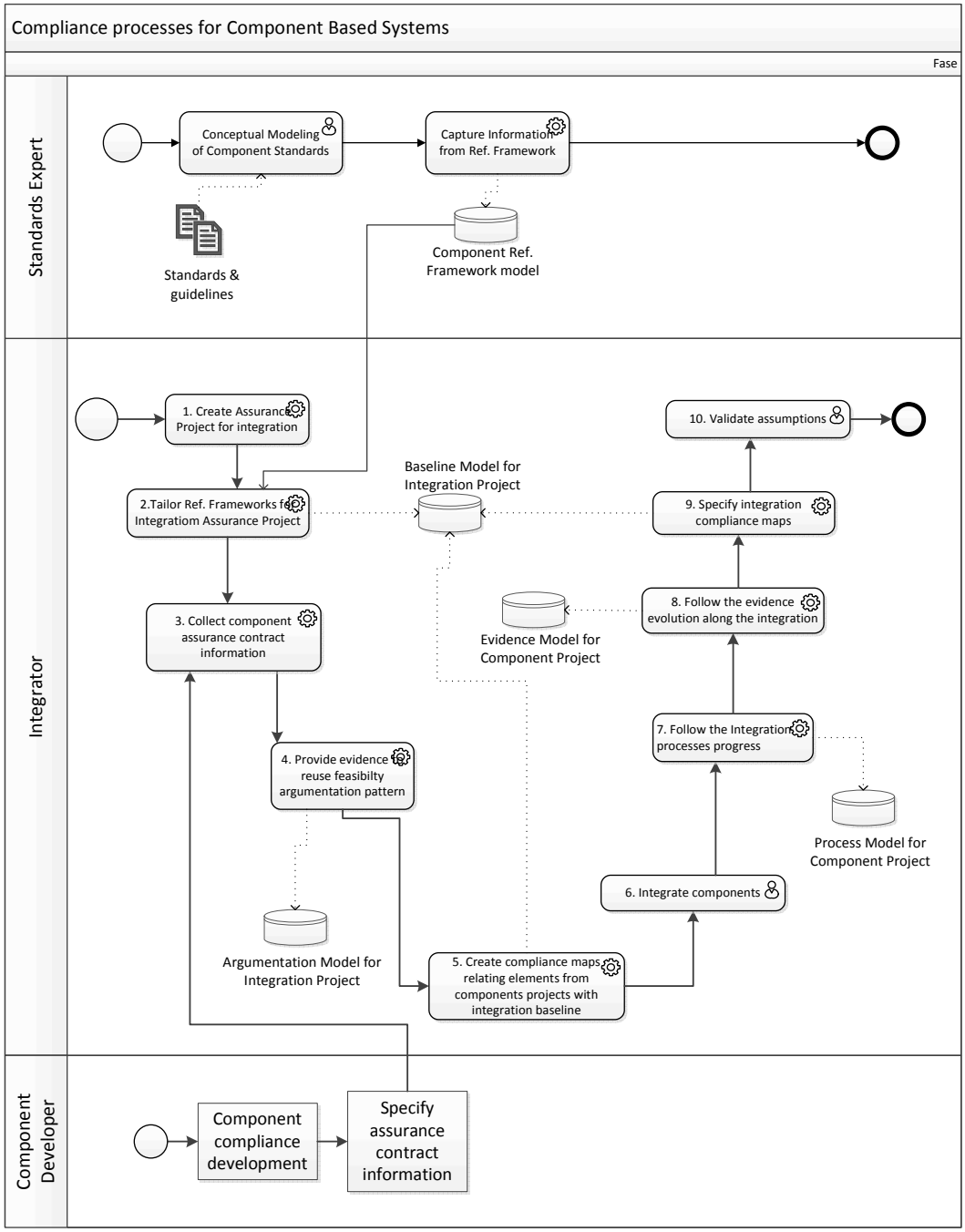


Fig. 61 Components integration compliance process

In managing complex industrial projects, it is a common practice to decompose the project into different subprojects for different teams to develop. Each of these teams will focus just on one of the subsystems in which the whole project is decomposed. These groups can be from the same company but they can also be part from different companies which are suppliers for the manufacturer. Each of them should see their

development as a project they need to assure compliance, it is an assurance project as we have defined on previous section. In this sense we will have a hierarchy of subprojects for each of the parts or component of the system. A subproject might focus just on the development of a software component or just on the hardware of a component. We can use this divide-and-conquer approach to manage the assurance of an overall system using this approach. We need a project of the systems and as many subproject as parts the system is decomposed in. In the same way, an "integration/reuse" is needed. It is necessary to open up another project to manage assurance aspects of the integration of components. This new integration project will 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. This is done by the integrator during steps 1 and 2. First the integrator creates an integrator assurance project and then the integrator will, tailor the baseline for the integration project.

For each component we should create a dedicated assurance subproject. On 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 done on other subprojects so we should address that on the project evidence model.

All this approach can easily be sustained with the meta-models we have described on previous section. 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 particularities that the baseline should include all of the post-conditions which it is necessary to meet for a complete integration.

For each of the components we need some contract data relating to a component needs to include 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 in step 3. Contract references to the artefacts and their properties, and the rest of the information from the artefact model is considered as a black box.

Integration assurance project should also include an assurance case focused on the actual safety integration of the components and the adequacy of the component reuse; this is the objective to fulfil in step 4. Components are safe standalone and that is what we will include on each of the components assurance project but on the integration assurance project we should focus on the integration and the actual verification that no unintentional interference occurs. For the reuse feasibility we have created an argumentation pattern in order to show the feasibility for component reuse. In Fig. 62 this pattern is illustrated. 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 on the same environment and at the same operational conditions and it if not the same it should be compatible.
- Compliance to standard: this aspect refers to standard objectives fulfilment by the reuse, showing requirements compliance to the standard requested on the environment where the component is going to be reused.
- Criticality levels: this aspect is referring to up to which level of criticality is the component developed. In safety-critical systems, functions are associated to hazards and the level of severity they could end up in case of occurrence. Based on this critical level, the level of demand to requirements coverage might differ. Components should only be reused into environments in which the critical level they are required is the same the one they were developed up to.
- Business case: on 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 worth the cost of the component.

In step 5 we deal with the compliance maps. Compliance maps, as described in previous section 4.6 Compliance maps, are the mechanism to indicate how exactly we are complying on a specific project to the reference requirements. In the integration assurance project we should reference to activities, requirements, roles, techniques that took part on the development of the component and are what we have called pre-conditions for the integration. We define preconditions as those constraints the component should fulfil in order in order to be reused. These pre-conditions can be either activities to be performed prior to the integration, artefacts that should be released to the integrator or claims about properties that should be ensured at component level. In the baseline for the integration assurance project we will include all the post-conditions which are all the activities requirements, techniques or roles that we have to fulfil during the integration and once the integration is done.

During the integration in step 7, the integrator should monitor all the activities that are performed during the integration. Some of these activities might have been specified as post-conditions. Some example of these activities is the execution of the analysis specified by the component developer to be executed in the target environment as it is mentioned in AC 20-148.

In step 8 in the integration assurance project we should add an artefact model where we all the evidences related to the integration appear and we also should include those evidences that are composed of parts which are components evidences. For example, the system safety plan should include all the components safety plans. In this sense in the integration assurance project, we will have the artefact model where we reference to the safety plans for all the components.

All these aspects are linked to each other and a change in one of them will impact on the others. In step 9 we should create the new compliance maps regarding the activities, artefacts claims about properties regarding the integration and at system level.

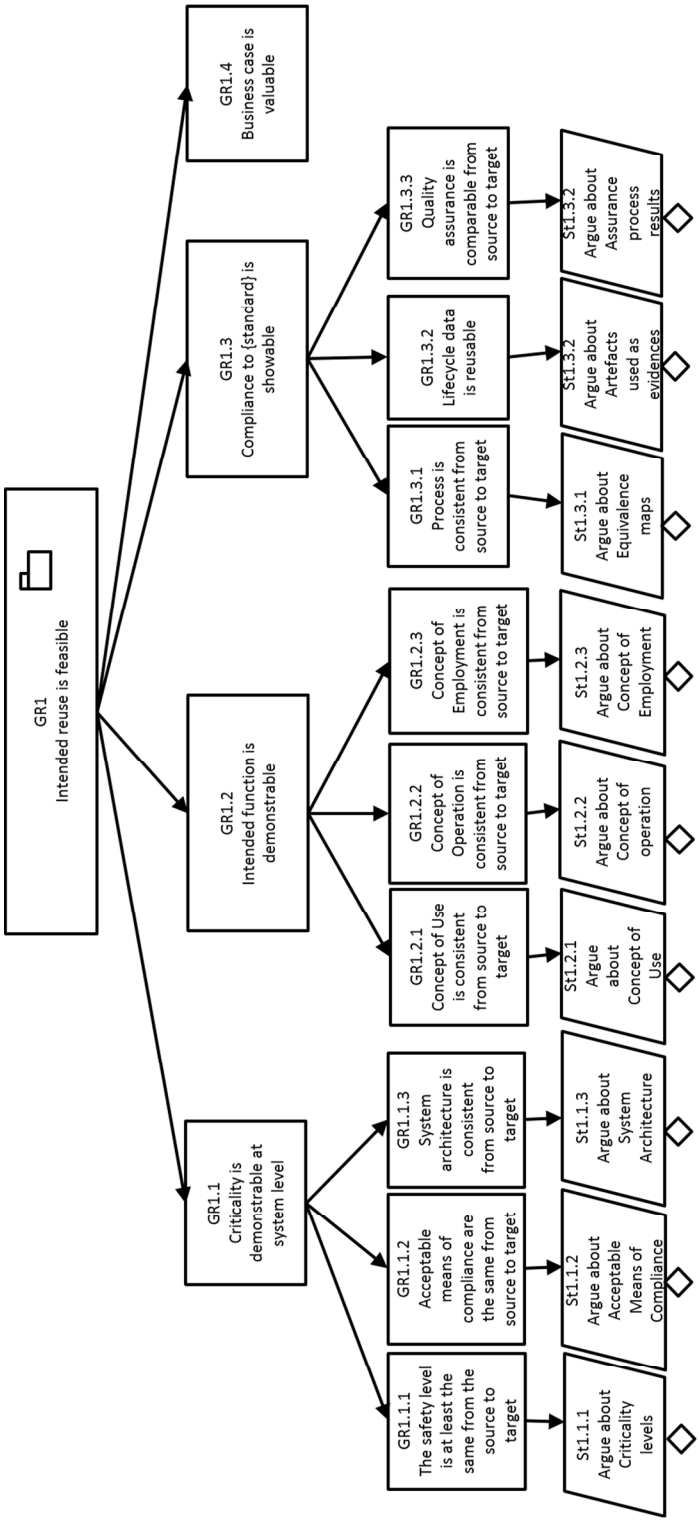


Fig. 62 Argumentation pattern for reuse feasibility

When validating a contract, it is necessary to take into account not only component functionalities and guarantees but also the context inherited by the components integration are compatible with the system context, and the assumptions made at development are valid once the integration is done. Details on how to specify and validate these assumptions will be detailed in the next chapter.

5.4.2 Rules for compositional certification

A complex project should be decomposed into different subprojects that different teams will develop. Just as previously mentioned, these teams can be from the same company or from different suppliers and companies.

Each of this team can be responsible of a subproject. A subproject might focus on the development of a component but this is not a limitation.

For integration of the work developed by different teams on different subprojects, an integration subproject is needed.

A project, “mother project”, should be the responsible off all the subprojects in which it is decomposed. This main project will take care of the assurance at system level. This project is called the “mother project” as all the subprojects are decompositions of this one and so they can be seen as sons-daughters of this mother project.

The integration project could also be the mother project if it also considers the system level compliance requirements. Sometimes it might be important to have these projects differentiated if the system level compliance requirements are too many to be considered in the integration or if the responsible for the system and the integration are different teams.

Each subproject packages a baseline model, an argumentation model, an evidence model and a process model. Compliance maps are also part of the aforementioned package. This is basically what we have described on the previous section about the meta-models required to model assurance. In this case this package will reference to information regarding just to the component.

A subproject/package could reference to multiple reference frameworks. It is clear for example in the case of an avionics reusable software component which need to comply with the DO-178c standard which is required for software developments on avionics and AC 20-148 advisory circular which is a guideline for reusable software component . In this example the component should reference to these two reference documents. The way to this can be to have one baseline per standard or reference document or either have a unique baseline combining all the references needed. For the first case we should have modelled both documents separately and have two reference frameworks to reference. For the second option we should have modelled in a unique reference framework both documents and the relations among them.

One reference framework can reference to other reference framework’s information like activities, requirements and/or artefacts. Following the previous example of the reusable

software component, AC 20-148 references to DO-178 lifecycle data, so in order to comply with the AC 20-148, the component should reference to artefacts (lifecycle data) which are part of other baseline, the one for DO-178 compliance. In the automotive domain, if we are considering that the hardware part of the system is one component and the software another component then we will have two assurance projects. The software component project will reference to an artefact which is shared by the two projects: the hardware project and the software project; this artefact is the hardware-software interface.

At subproject level we need to include as minimum one baseline in order to mark to which standard/recommendation we will comply with. Each subproject should be compliance as minimum with one reference document.

On a subproject, there can be as many baselines as necessary. At least there will be one baseline per reference framework associated. It has been previously mentioned, if we want to reference different standards we can have one baseline per each of them. Apart from that, we can also update a baseline if we need to refine or tailor the way the standard is complied. If we update a baseline with modifications, we should store previous baseline for tracing purposes. To distinguish if a baseline is an update or a previous version, we will only have active those up to date baselines. Old versions will be deactivated but stored.

Regarding specific interpretations, we will take care of those one at (sub) project level. A subproject level should be done by tailoring baselines. For example ,if on a project it is agree to use a specific tool used to cover specific requirements, for sample used a requirements management tool to cover the trace of the requirements, this is done at project level. Decisions of this type done at “mother project” level could affect all the subprojects if it is agreed this way but it is not necessary on all cases.

On a subproject we can reference to activities, artefacts or argumentations developed on another subprojects. For instance, when one team assumes that specific activities are done by other team. On automotive a system team assumes that the hardware team will make some verification activities regarding the hardware. This implies that a subproject can reference and have access to any other related subprojects public activities, requirements, artefacts or argumentations.

Evidence models of a subproject can reference to other parts of evidence models for other subprojects. In the avionics domain we have the artefact PSAC (Plan for Software Aspects of Certification) at system level should include information of a component/subsystem PSAC. In the automotive domain the hardware-software interfaced can be the responsibility and be managed by the hardware team and the software team will reference it on their evidences.

Evidence model of a subproject can be referenced from other subprojects in order to create compliance maps. Following the previous example the software subproject will reference to claim compliance with the standard ISO 26262 to the interface software-hardware that is created and compiled on the evidence model of the Hardware

subproject. At system level project we will also reference the hardware-software interface as it is requested at this level as well.

Referring to other subprojects activities or evidences, introduces the problem regarding confidentiality versus accessibility. If we consider components as black boxes with only the interfaces to others accessible, then all the activities done for component compliance might not be accessible. There should be a clear profile policy where depending on the user role the access to different elements should be allowed.

Claims for one subproject can use evidences from others subprojects to support the argumentation. This is the application of the notion of the away solution from the GSN standard or the argument element citation in the SACM standard.

Claims can be decomposed into sub-claims and these sub-claims can be developed into other subprojects. This is the application of the notion of the away solution from the GSN standard)

We can consider a contract as a collection of all dependencies between subprojects of a project. When a subproject references to an element of another subproject, then this dependency should be reflected in the contract.

There should be contract at system level with all the dependencies of the subprojects which are part of the mother project. In this contract, we should reference to the safe integration that is included on the argumentation from the integration subproject.

A contract should reference to process done (activities) compliance related concepts (reference frameworks and compliance maps), public evidence and argumentation, both public claims and integration argumentation.

5.5 Tool support for composition assurance

The tool support presented in previous chapter has been extended in order to support the integration phase described in this chapter.

For creating reference framework the process is the same as the one described in section 4.7.1.

After the creation of the reference(s) framework needed is time for the creation the system assurance project and all the subprojects in which it is decomposed.

The first step will be the creation of the “mother project”. To do that, we will use Assurance Project Wizard. There, we will define which standard(s) and requirements apply to the specified component as it can be seen in Fig. 63.

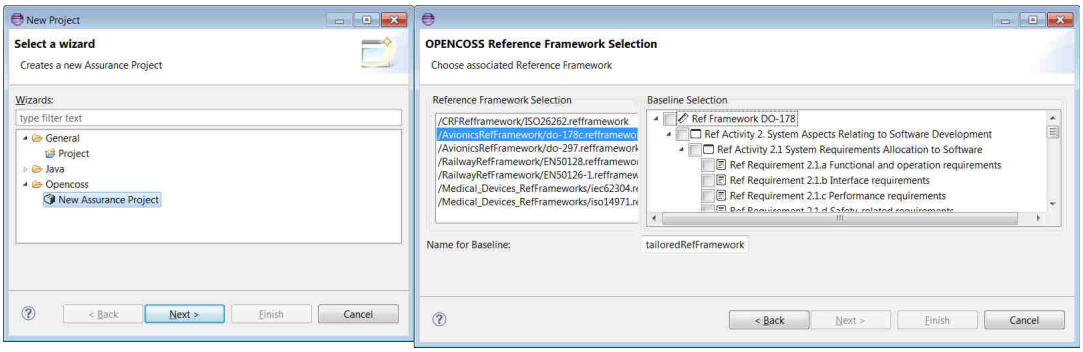


Fig. 63 Assurance Project wizard

Additionally, it is necessary to open up as many new Assurance Projects as components the systems is decomposed. In our example the mother project will also manage assurance aspects of the integration of components, for doing that we should define the component assurance projects as subprojects of this integration project. In Fig. 64 we see the assurance project hierarchy.

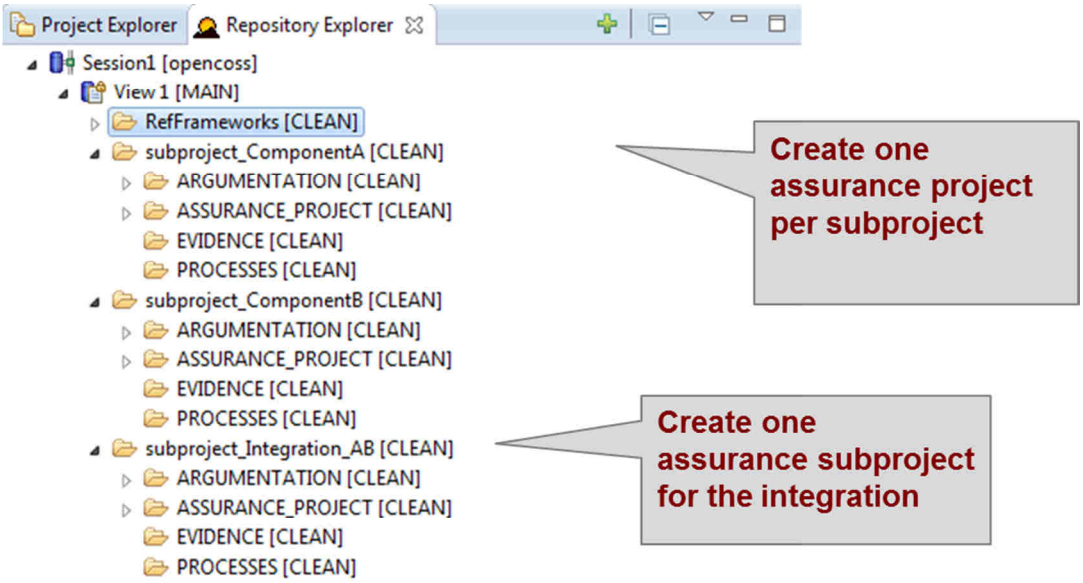


Fig. 64 Assurance projects hierarchy

We select the integration assurance project, in order to have access to its properties. One of the properties an assurance project has is the subproject. We will add all the subprojects which are involved on the integration.

For an assurance project (in this case, for a component), we should define the evidences that we will provide to support assurance onto the evidences model editor, Fig. 65 shows an example of evidences associates with robust portioning compliance.

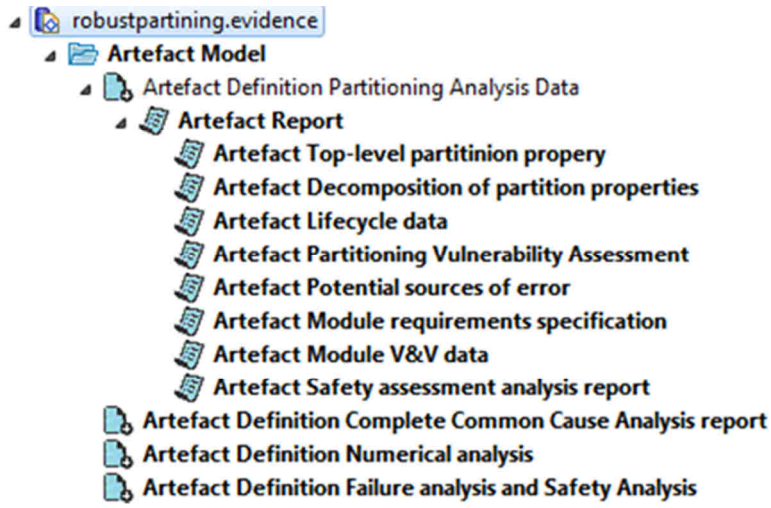


Fig. 65 Evidences Model Instantiation for Robust Partitioning Related Evidences

We will create assurance cases related to each of the component in each of subprojects. In Fig. 66 we can see the editor developed to support this creation.

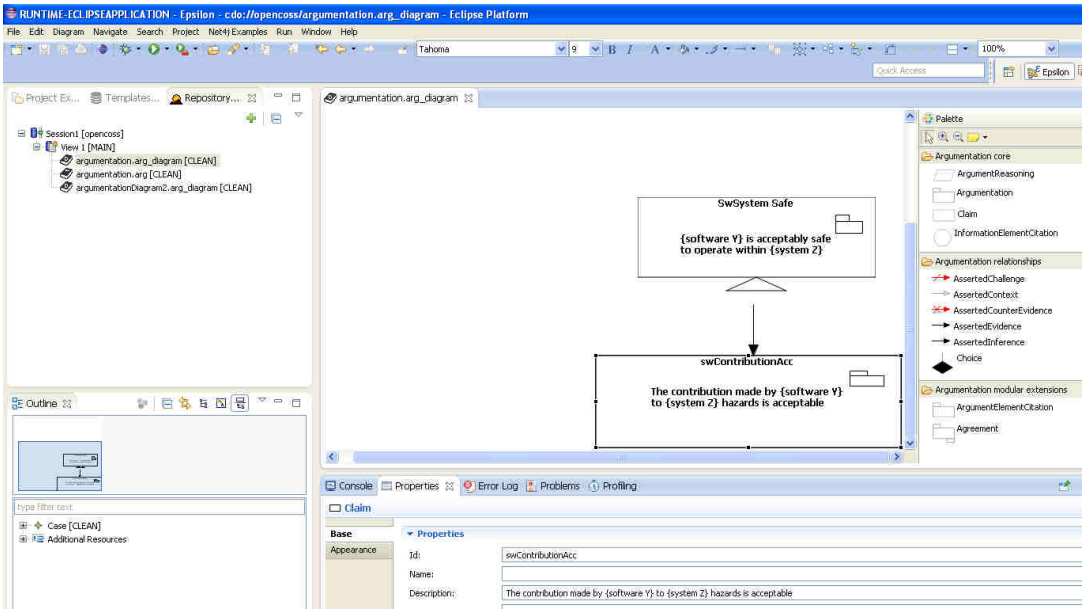


Fig. 66 Assurance case editor

In order to facilitate the task of creating the argumentation, the tools also provide support for the argumentation pattern use. In the argumentation templates view we can see the list of patterns stored on the platform. Just by drag and drop the needed pattern we can start instantiating and adapted to the actual component.

In each of the assurance project we will also create a process model to follow the project at all times and manage the execution of the processes. The process model editor is shown in Fig. 67.

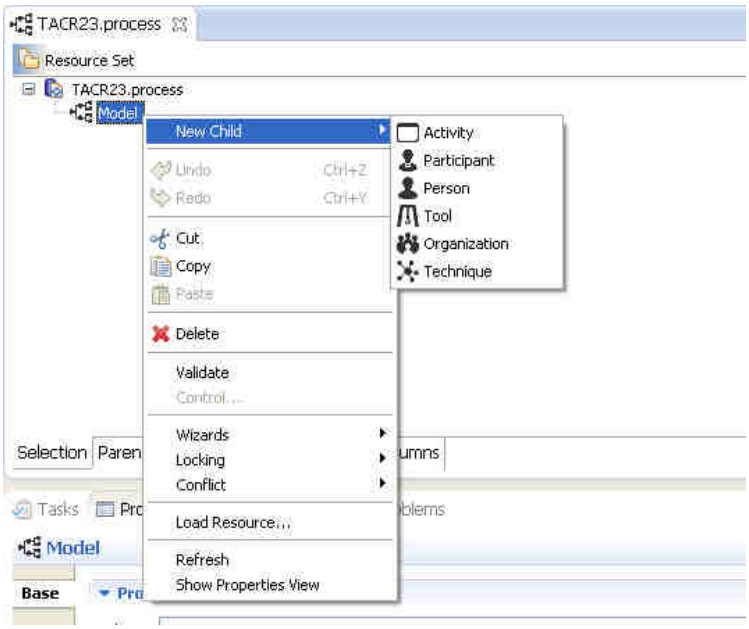


Fig. 67 Process model editor

We will need to create compliance maps as a mechanism to show compliance with the standard requirements for each of the subprojects. To create Compliance Maps, first of all, it is necessary to press the button “Mapping Set” on the properties form of the baseline of each of the assurance subproject.

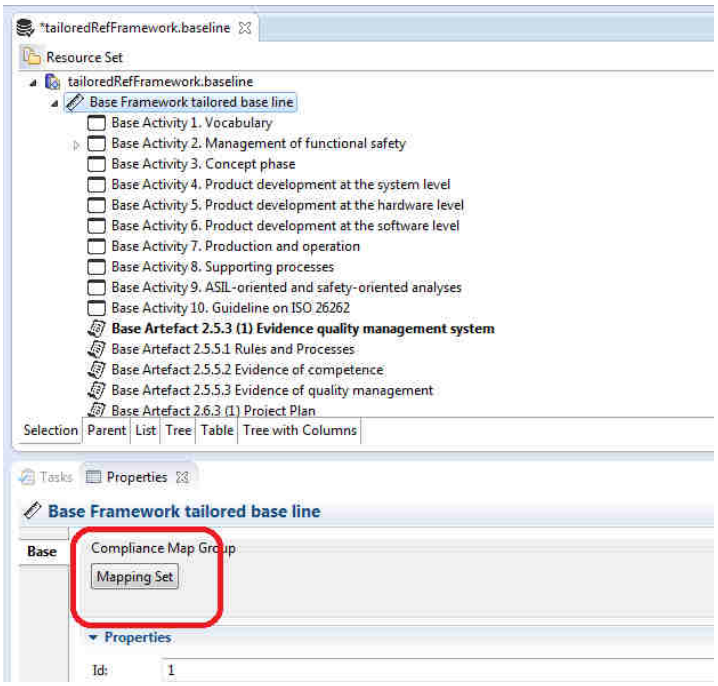


Fig. 68 How to create Compliance Map

The Compliance Map form is organized in three zones:

- The *left zone* shows the actual baseline, and it loads the type of elements for which we want to make the compliance maps. For default, activities.
- The *middle zone* allows to make different filters like:
    - *Filter Mapping Model* lists all the mapping models stored in the database, and it will be necessary to select one of them and one group model. It's also possible to create a new map group pressing the button "New group". This map group has to be part of the active Baseline Config of the project.
    - *Filter Map Element*. It's possible to create compliance maps for activities, artefacts, requirements, roles and techniques, and the allowed maps are:
        - BaseArtefact -> Artefact
        - BaseRequirement -> Artefact , Claim or Activity
        - BaseActivity -> Activity
        - BaseRole -> Participant
        - BaseTechnique -> Technique

      When the filter changes, also the information showed by the reference framework changes.
    - *Filter Compliance Map*. This filter allows making different compliance maps for the same element.

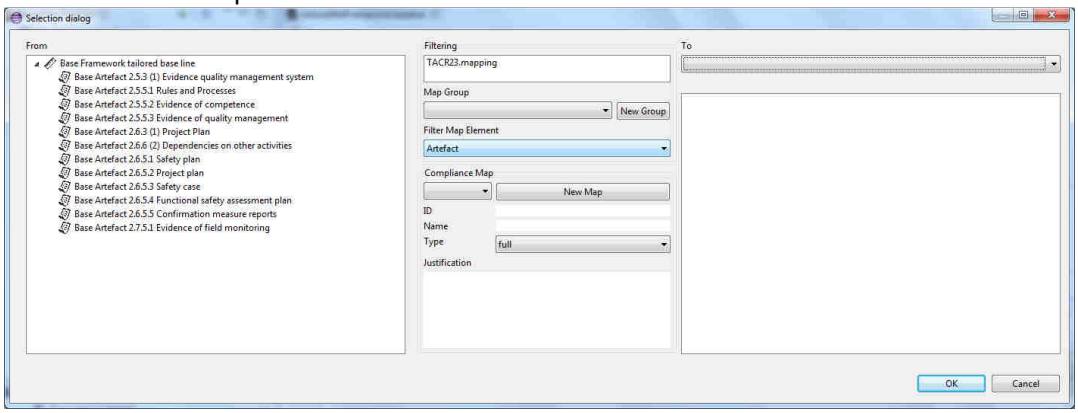


Fig. 69 Compliance Map, select map element

The *right zone* shows the list of models; depend on the map filter selected. We should select one of them. This selected model will be the target of the compliance map to create.

In the integration baseline we should include all of the post-conditions which it is necessary to meet for a complete integration. For example all the activities that should be performed, once the components have been integrated. The argumentation section of the integration assurance project will focus on the actual safe integration of both components.

In the integration assurance project we should also define compliance maps for those concepts that have been included on the baseline. We are able to reference activities, evidence and parts of the argumentations which belong to the subprojects.

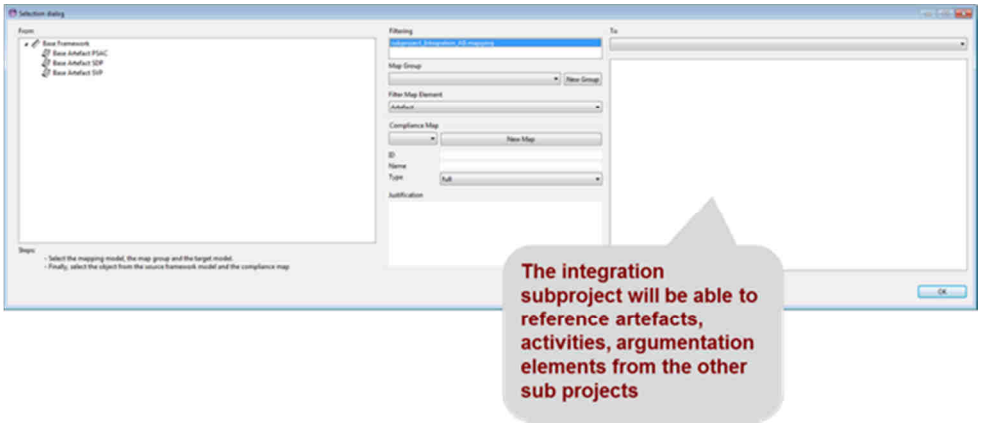


Fig. 70 Compliance maps at the integration assurance project

We just have to follow the same procedure as with the subprojects but on the integration project we will reference to activities, evidences or claims that are described on the different subprojects.

Apart from these editors we also need to check the status for compliance at any time of the project lifecycle. For that need, a compliance report feature has been developed inside a server letting users assess the current compliance of their project to the selected safety standard.

The functionality is intended to be used by:

- Project team members, such as developers, when the project is in progress, in order to have up-to-date insights into which of the baseline framework items are already satisfied and to what extent.
- Project safety manager in order to monitor the project general compliance, observe the compliance details and add, assign, or un-assign specific evidence resources to/from the given requirement of the safety standard which is followed by the project.
- Independent safety assessor, when the project draws to an end, in order to browse the assigned safety evidence, evaluate it and independently assess the actual project compliance to the specific safety standard.

In Fig. 71 we can see an example of the online version of the compliance report.

The “Project Compliance” table, which is placed on the left, presents base artefacts and base activities of the selected safety standard. The most important column is the “Compliance Status” one, which presents the overall compliance status of a project to the specific safety standard item. The column can be sorted by value, thus allowing user to assess the project compliance at one glance.

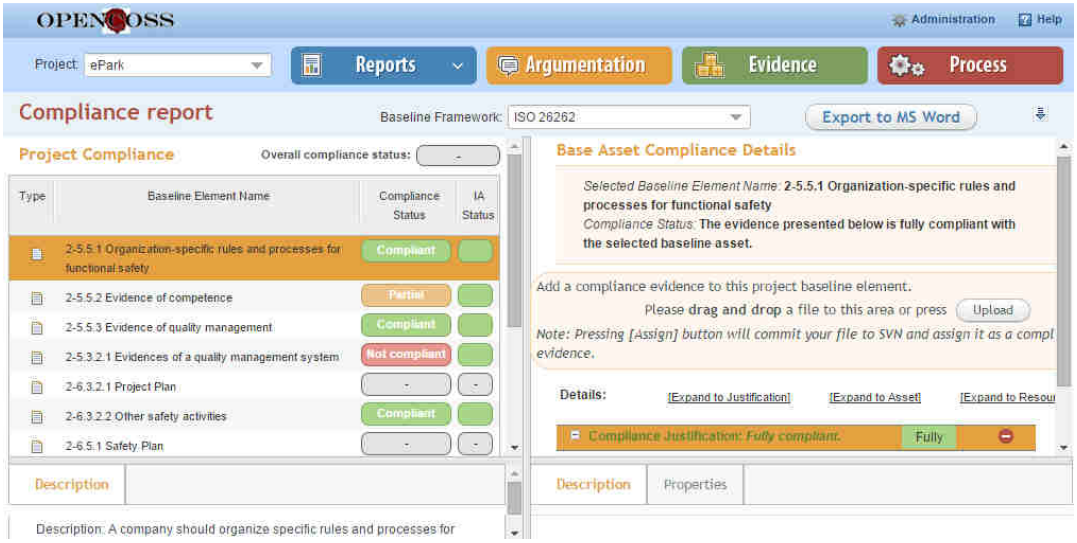


Fig. 71 Online version of the compliance report

Internally the compliance report feature analyses the project to check the compliance maps created and shows a summary of the status of all the compliance maps in relation with all the compliance obligations defined on the project baselines.

5.6 Different applications for the Composition approach

In the standard analysis section we have mentioned different concepts accepted by standard which maps with the idea of compositional assurance.

In this section we will show some of these examples extracted from the standard and from a high level perspective describe how the approach presented on this section is applied. For a more detailed demonstration we should reference to the case study section of this thesis.

Avionics domain section

In the avionics domain we have mentioned the IMA architecture as the modular reference platform. For this example, we will think on the sense and avoid function. This is a critical function that allows pilots to detect whether there are any possible collisions on the way and defines the most adequate respond to avoid them. This application function can be reused from different airplanes or even from UAV (unmanned avionic vehicle). In this case we will define this as our reusable application.

This avionic function should run on a platform and as mentioned before in this case we will be allocated into an IMA platform. Inside this platform different modules will run the basic software to make the platform run. In our example we will highlight one core IMA module which is in charge of the partitioning services.

When analysing this example we detect three teams working in parallel:

- Sense and avoid function team

- IMA platform team
- Partitioning module team

Above these three teams, a fourth team is waiting to integrate all their results.

If we extrapolate the approach describe to this example we have four assurance projects, one per team. The integration team will be considered as the “mother project” which will take care of the integration assurance and system level assurance. This mother project will consider the other projects as its sub-projects or children.

A variant for this example could be to have the IMA platform team and the partitioning team as two subprojects and the IMA platform team will also take care just of the integration of the partitioning services with the IMA platform. Then in parallel we will have another integration project just focusing on the integration of the IMA function (the sense and avoid) with the IMA platform.

The sense and avoid function will have two baselines, one associated with the DO-178C compliance and one with just some part of the DO-297 standard, just the requirements for IMA functions reuse.

Similarly the partitioning module will also have 3 baselines, one regarding the compliance with the ARINC 653 (Avionics Application Standard Software Interface) which specifies some requirements for the interface the module should comply; the DO-178C as it is a software development and some parts of the DO-297, the parts about reuse IMA modules and also some specific requirements regarding partitioning and isolation.

Automotive domain section

In the automotive domain we can have the idea of the SEooC, the qualified software or the qualified hardware as possible concepts for the system decomposition. For this example, the system is a new full electric car. This system is decomposed into different systems, but we will focus on two systems, one is the parking system which is going to be developed by a complete different team as a system SEooC.

The responsible of the car will see the parking system as a black box and only some information will be released to the integrators. On one hand the assumptions made at the developments will be accessible along with all the work products requested by the ISO 26262 for the system SEooC.

We will also have three more teams working on the car; all of the teams will be working on the powertrain system. The powertrain system will be decomposed into hardware and software, one team will work on hardware and the other team will work on software. A third team will work on the integration of hardware and the software from the system perspective, The software team will only work compliance with part 6 of ISO 26262, the part that deals with software. This team will reference to work products done at system level such as the system technical requirements; they will share a work product with hardware team; that is the interface hardware/software. The hardware team in a similar way will focus on the compliance of the part 5 of the standard, which is the part that

deals with the hardware requirements. The integration team for the powertrain system will deal with part 4 of the standard and reference the results of the other two teams.

There should be an integration project that will see all the previous projects as subprojects and will deal with the integration of the parking system and the powertrain system with the vehicle. This project will include on its compliance activities one more which is the validation of the assumptions made during the SEooC development.

To sum up we have the following subprojects:

- Powertrain system hardware development team
- Powertrain system software development team
- Powertrain system integration team
- Parking system SEooC development team
- Vehicle integration team

Health domain section

In the medical devices sector there is a concept for reuse which is the SOUP, software of unknown procedure. In our example, we have an automatic external defibrillator (AED) as the medical device to develop. Inside the AED we have two components that are being developed by two teams, the signal analysis and the shock generator, for each of them we will have an assurance project. And we also need the AED system project which will act as the “mother project” and will also take care of the integration of the two subprojects. For the integration we will treat the outputs of the subprojects as SOUP, with the same level of requirements. Medical devices should comply with a great variety of standards. In our case the “mother” project will have to comply with ISO 14971 which addresses the risk management and it will also address the IEC 60601 for the electric safety. However, ISO 62304 which is about the software development will be addressed by the other two teams which are the responsible for software development fir the device. In the subprojects then the standards IEC 60601 for electric safety and the IEC 62304 for software development will be addressed.

5.7 Conclusions

In this chapter we have shown how the components are reused and integrated in a complete system. The proposed CCL presented in chapter 4 is adapted in this chapter to be used for composition. The concept of subprojects and the new integration assurance project is important to understand the contributions of this chapter. The component assurance project that was described in chapter 4 is used here as a subproject for the system assurance project. The integration assurance project will reference to them and integrate their activities and artefacts into the integration baseline. We have introduced the idea of the baseline for integration to ensure the activities, artefacts and requirements for the standards and guidelines are managed. The concept of compliance maps is used in the integration assurance project to as a mechanism to trace compliance.

The argumentation pattern for reuse feasibility is also introduced in this chapter. This pattern takes into account elements for compliance, for safety and also for business strategy.

We have intentionally left out of this chapter the validation of the assumptions made at component level as this will be described in detail in chapter 6.

In Table 10 we have summarized the issues and challenges which were identified at the beginning of the section and how we have tried to give them a response.

Table 10 Conclusions to challenges identified in chapter 5.

Id	Challenge	Challenge Description	Mechanism	Solution description
1	Responsibility decomposition	When decomposing the system into smaller parts usually developed by different teams and even different suppliers, responsibilities are hard to assign. Whole system assurance is not possible to be accomplished without the support of component assurance.	Assurance Project hierarchy	We propose to have a component assurance project for component assurance and a "mother" project which will take care of full system assurance. On each of the subprojects the responsibilities are assigned.
2	Need to reference to others' activities	Components are developed in an isolated way, however, on their assurance processes they do need to reference other components or the system development activities	Baseline	On each of the component assurance projects we will create baselines indicating what activities are planned to be performed at component plans to comply with the standard. These activities can be referenced from others baselines, from example from the integration assurance project's baseline

Id	Challenge	Challenge Description	Mechanism	Solution description
3	Reference to others' evidences	Not only a component need to references others team's activities but also evidences. Results from analysis wouldn't have any sense without the context, or constrains defined and verified on others component.	Compliance maps	On each of the component assurance projects we will create evidences model indicating the evidences generated at component level. Other projects could use the mechanism of the compliance maps and reference other projects evidences in order to fulfil their own requirement.
4	Evidence refinement by composing artefacts from different components	Evidences from a component are necessary to refine artefacts used as evidences at system level. In this sense safety analysis like a component fault tree analysis is required to complete and refine safety analysis at the whole system level.	Evidence composition	At the "mother" project, we will create an evidence model that can have access to all the components evidences model. Other evidence models from other projects will have access to these evidences and reference them on their own evidence models.
5	Need to show integration from the assurance perspective	It is important that component integration is done not only at technical level but also assurance should be cared	Integration assurance project	The integration assurance project is the responsible to take care of assurance for integration related issues. We are able to see the integration compliance by looking at the compliance report of the integration assurance project

Id	Challenge	Challenge Description	Mechanism	Solution description
6	Capability to trace what each team is doing regarding assurance	For the system responsibility point of view it is important in order to see the progress of the whole project, the fully accomplished system, how each of the team is going	Compliance Report / compliance maps	At any time we are able to see by accessing to the compliance report the status of the assurance projects. Internally the compliance reports will check the compliance maps created at each of the assurance projects

"Imagination is more important than knowledge." — Albert Einstein

6

Contract-Based Assurance

6.1 Introduction

This chapter is focused on the contract-based approach for components assurance on safety-critical systems. Contract-based approaches are seen as a common and well know strategy while dealing with complex systems. As systems have grown in complexity, so does the trend in using contract-based development approaches to deal with components integration.

Contracts are defined in [Benveniste et al. 2012] as the agreement between an OEM (Original Equipment Manufacturer) with its suppliers on the subsystem or component to be delivered. “Contracts involve a legal part binding the different parties and a technical annex that serves as a reference regarding the entity to be delivered by the supplier—in this work we focus on the technical facet of contracts. Contracts can also be used through their technical annex in concurrent engineering, when different teams develop different subsystems or different aspects of a system within a same company”.

Contract-based approaches differ when we see them from the development perspective, from the safety perspective or from the assurance perspective. Benveniste proposes contracts for design to be used at nearly all stages of system design, from early requirements capture, to embedded computing infrastructure and detailed design involving circuits and other hardware. Contracts explicitly handle pairs of properties, respectively representing the assumptions on the environment and the guarantees of the system under these assumptions. The component is assumed to have a correct functionality just by ensuring the interfaces with others are compatible. From the safety perspective, assuring the correct function of components does not mean that the (composed, integrated) system will remain safe. From the assurance perspective the component not only need to be safe but also should comply with the standard requirements. Just as safety, assurance is evaluated at whole system level, in case of the avionics domain, the assurance is done at aircraft level, in the automotive domain at vehicle level and in the health domain, and we look at the complete medical device no matter if these are a set of systems working together.

In chapter 5 Compositional Assurance Approach we talked mainly about how assurance should be done at system level managing all information which is extracted from the different components that form the systems. However, we did not mentioned

information about the properties that should be assured on system level and how this information should be treated.

This chapter shows a structured way to describe the assurance contract approach in order to reduce ambiguity and creating the first step for tool support on managing contract information.

This chapter is organized as follows. First the challenges section introduces the main problems we are facing when creating assurance contract, then we distinguish between the different types of contract we can see on a safety-critical system. Following there is a context literature review on the existing approaches regarding contract approach. Afterwards we talk about the structure for contracts and the structured expressions proposed for each part of the contract. Next section is about how the contracts evolution through the system development lifecycle and what is their purpose on each phase. Finally we describe some conclusions and resume how we have fulfilled some of the issues described at the beginning of the section.

Fig. 72 highlights the scope of this chapter in relation with the compliance processes for component-based systems. This chapter impacts in the activities done by the component developer and by the integrator.

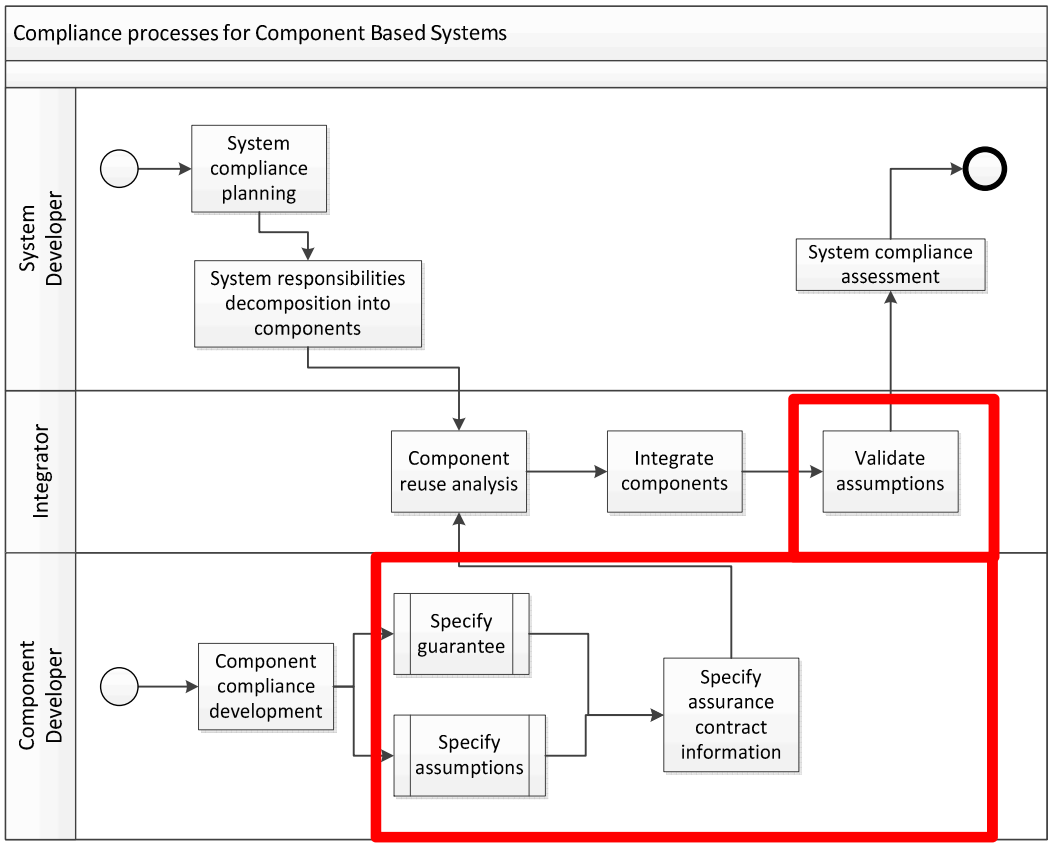


Fig. 72 Scope for chapter 6 in the compliance processes for component-based systems

6.2 Challenges

In 2013 there was a request to the OMG in order to define a standard for Machine-checkable Assurance Case Language (MACL) [MACL 2013]. The intent was to solicit information for standardizing an abstract syntax and semantics for languages that describe construction methods of machine-checkable assurance cases. Information sought includes what technologies exist for checking integrity of assurance cases, what features of assurance-case description languages and notations enable those technologies, and how those features interact with other features of the languages such as abstraction and modularization. As they say their objective was to define: "A language to describe assurance cases and more generally their construction methods has potential to enable mechanical checking that replaces mundane and not-so-mundane parts of those manual reviews. Its role will be similar to a strongly typed programming language: type checking frees programmers from worrying about type-errors that can be subtle and burdensome to check manually; avoiding them is only a part of the problem but type-safety can guarantee a great deal when used wisely."

There are some identified reasons behind formalizing text, and multiple ways of expressing rely/ guarantee conditions. [MACL 2013] provides the following examples:

- Avoid human errors
- Support for validation or checking
- Interoperability between different actors from a system supply chain" (including OEMs and component providers)
- Facilitate the integration of the components within the system

Human factor

The human factor is a very important topic in safety-critical systems. People get tired, distracted, and they can omit some details. System are becoming more and more complex, this complexity is being managed by decomposition into components however, the assurance should be handled at system level and this complexity could affect those in charge of assuring the whole system. The Nimrod review [Haddon-Cave 2009] is a report demanded by the UK ministry of defence to wider issues surrounding the loss of plane Nimrod XV230 in Afghanistan on 2 September 2006. There it was pointed out that: "Complexity is normally the enemy of Safety and the friend of Danger".

In the previous chapter the requirements from the standard in relation to composition have been analysed. We have extracted the properties that should be checked during the component development and after they are integrated (See ANNEX A). Having just an overview of them we discovered the heterogeneity of the properties. We have grouped them by categories as: installation, safety, operational, functional, performance and reuse. This classification came from the avionics guidelines AC 20-148 [AC 20-148] which identifies the first four categories as the topics that stakeholders should identify any possible concerns that could came up when reusing a software component. We have added a new category, the reuse category for those properties which do not fit on the other categories but needed to be assured on the components so they could be reused.

Validation and checking

One of the benefits of formalizing safety contracts will be the possibility of tool support for checking or generating contracts. The creation of a formal grammar that information inside a contract should follow could supports the technical approaches and instantiate the guidance from the standards. Moreover, with the provision support of a defined grammar for safety contracts we will be able to support validation of contracts (e.g. helping identify incomplete contracts).

There is a need for a checklist so as the person in charge of assuring the system takes into account all the information which is provided in a simpler way. Depending on the perspective and information a person needs at some point, different information from the contract is relevant. In [Ruiz et al. 2013-2] we say: "viewpoint will let us handle the different aspects in a unify framework, this way different type of contracts in a common and systematic way structuring the information and this way helping to assure completeness. Managing contracts may be complex but with the suggested approach, we will give a process for component composition a structure, making it more manageable and linking safety behaviour with safety properties".

Interoperability between different suppliers

As mentioned in the previous chapter there is a need for a uniform way to express what a component is offering and what a component expects from others. This unique way to express this is especially important for the different stakeholders. We need to show that are the demands from the components from other parts of the system that while the components has been developed this demands have been assumed as fulfilled demands. We are also requested to express the guarantees the component offers. We shall illustrate this on a uniform way to the different people which participate on the system development, so the integrators and other component owners knows what is expected from them and what can they ask to that particular component. The unique way would benefit all stakeholders as information will interoperable among all.

Facilitate integration of the components within the system

Having a structure and a clear way to define the interfaces of the component from the assurance perspective will benefit the integration. At integration phase we have seen the contract-based approach for technical integration for some properties such as timing. These approaches have been valid for complex system integration but their use from the assurance perspective has not been extended.

Assurance is a system property and although it is difficult to decompose on the different components, with a contract-based approach we could benefit from this approach. Contract with assurance information for the properties will let us verify its correctness for a safe integration. If the information is provided in a formal or semi-formal way, then some tools could be later on applied to confirm the validity of the contract.

Learning curve

Specifying contracts in a formal way is not widespread in the industry due to the difficulties in learning a formal language. Formalizing contracts should be done in a way that it is easy for the developer and integrator and supported by tools, other ways it will be difficult to be adopted.

6.3 Analysis of Contracts use

Contracts have been used on safety-critical systems for a long time and for different purposes. One of the main problems that arise here is the naming problem. Contracts are used for different purposes depending on the interest on the people who are applying it. When people with different roles and/or backgrounds come together and decide to apply a contract approach, their interests on the results differ. In this section we have categorized the contract-based approaches depending on the purpose. The main interest here is to disambiguate the term.

We have identified three phases in the use of contracts to support the certification of components:

- Design contracts
- Safety contracts
- Assurance contracts

Design-by-contract

A contract-based design in system engineering involves the following cornerstones:

- Components and subsystems
- Contracts
- Interaction points

We can define design contracts as those agreements made for development purposes where interfaces between components are identified and agreed in order to facilitate the interoperability and integration of components. Design contracts have as one of their main challenges the interfaces within other components, so the integration is done correctly. The component is assumed to have a correctly functionality just be assuring the interfaces with others. From the safety perspective, safety is a whole system property and just by assuring the correct function of one components does not mean that the system will remain safe, as that precise component could have impacted in the behaviour of another that with the integration is unable to get to a safe state in case of an event. The first step is the use of design contracts to support the technical integration of different components within a system. Design contracts focus on the necessary conditions for correct component operation. In an integrated component configuration if component contracts are satisfied the set of components can be assumed to function correctly together.

Safety-by-contracts

For this type of contracts we focus on those characteristics and properties that need special attention regarding safety. Behaviour is one of the characteristic that SPEEDS [SPEEDS D2.5.4] first and then CESAR meta-models [CESAR D_SP1_R3.3_a_M3] did take into account. However, behaviour for those meta-models is seen as nominal function behaviour, reaction to its environment. Abnormal behaviours or behaviours when failures occur where ignored but are essential from the safety perspective. Safety is a system property so the task of allocating the safety properties to specific components is difficult as the responsibility is shared by all the components. The context in which the component is going to be integrated is important and as we [Ruiz et al. 2013-1] indicated for the SEooC perspective the assumptions of the item can be understand as the context characterisation. In addition, to support safety assessment, failure behaviours of components, and their behaviour in the presence of failures, must be defined. Ruiz shows some needs of the industry in relation with the application of the SEooC concept and proposed the use of safety contracts as a possible strategy. 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. The core of those safety contracts will be the characterization of the context in which the components is expected to be integrated with and the possible behaviours of the component. Understanding of the safety decisions made at design time are important while analysing the different environments in which the component can be integrated as it support the impact analysis required in case of a discrepancy of the context.

Assurance contracts

Assurance contracts define the set of claims that need to be made concerning a component to support its certification against a particular safety assurance standard. Different standards address this problem in different ways. In ISO 26262 [ISO 26262] Development Interface Agreements (DIA) are described as a way to specify both procedures and responsibilities allocated to distributed developments for items and elements. The DIA includes information beyond technical safety by addressing procedural and confidence related issues. The use of DIAs is intended to help address risks such as: a supplier with inadequate capability, improper understanding or definition of the boundary of component and its interactions with its environment, or failing to fulfil requirements.

Regarding those assurance contract, different standards has focus the problem in different ways. An example of the contract could be seen in the ISO 26262, where the term Development Interface Agreement (DIA) is used to define the procedures and responsibilities allocated within distributed developments for items and elements.

In the avionics domain we can find similar requirements when talking about modules and application reuse on an IMA (Integrated Modular Avionics) platform. On the DO-297 [DO-297] (amongst other requirements) it is required that limitations, assumptions, etc. are

documented and a usage domain analysis performance to ensure that any component is being reused in a way that is compatible with the original design intent.

In the avionics domain the context is already known as the IMA application can only be integrated inside an IMA platform and the specification of the platform is well known so technical information for the integration is not discussed. But in the avionics domain, adequacy of the supplier such as the DIA requested on the automotive domain is also a big concern. Big companies such as AIRBUS are starting to put into practise a methodology to ensure the quality and capability of their suppliers especially for the critical functions. Anne Yani presented [Yani 2011] the plans for Airbus on the idea of extended airworthiness. The main certification constrains for EASA on this topic were:

- Delegation of authority
- The cascade on certification requirement
- The surveillance of suppliers

Airbus has a strategy were principles and means are validated with EASA, defined the supplier contributors for the certification process and different stakeholders' feedback can serve for improvement. In order to put this strategy into practice Quiniou [Quiniou 2011] presented a methodology where AIRBUS classifies their suppliers depending on two aspects:

- Severity as design: Level of risk of the engineering activities based on their impact on Aircraft) and
- Confidence in supplier : Assessment of the supplier organization and associated supply chain including sub tiers to deliver to their contractual obligations

Suppliers are aware of the certification process and take part into it, in order to do it responsibilities and work need to be clearly defined in order to integrate the different suppliers' products into the system.

The proposal from [Ye Kelly 2004] Ye and Kelly where use of a contract "must record an account of the match achieved between the objectives required by the application argument module and addressed by the COTS component argument module. In addition the contract must also record the collective context agreed as consistent between the participant modules" has inspired this work. The contract-based assurance approach proposes the use of contract not just for COTS but for any component included in a system. The contract shall highlight the context in which the resulting system from the component integration will work in. This context is defined in form of linked assumptions and guarantees between the components and the system.

SAFECER project has been analysing the possibility to provide support for system safety arguments based on properties of system components that can be extracted from the system models. Sljibo [Sljivo et al. 2013] proposes a contract extending the approach of the basic contract form of assumptions/ guarantees in a form of assertion by broaden the scope of the assumptions. They propose "to include not only the component environment in terms of other connected components, but also usage context (e.g.

frequency if a service usage), hardware context (e.g. available memory), development context (e.g. compilers) and system context. This work takes their idea of assertion forms for assumptions and guarantees specifications.

Rushby [Rushby 2010-2] proposed to have formalised the claims in the safety cases. Formalizing some elements of the safety argumentation will support precision and could be used to apply checking methods like SMT solvers He does not provide a complete formulation proposal but just some examples on the feasibility to do it. We have followed this approach by proposing structured expression for the claims in the safety cases that are used as assumptions or public goals.

We have mentioned in chapter 2 the initiative at the OMG, where there is an ongoing discussion for the need to create another standard, the MACL (Machine readable Assurance Case Language) in relation with the SACM standard [MACL 2013]. In Fig. 73 we can see the relation between the two standards.

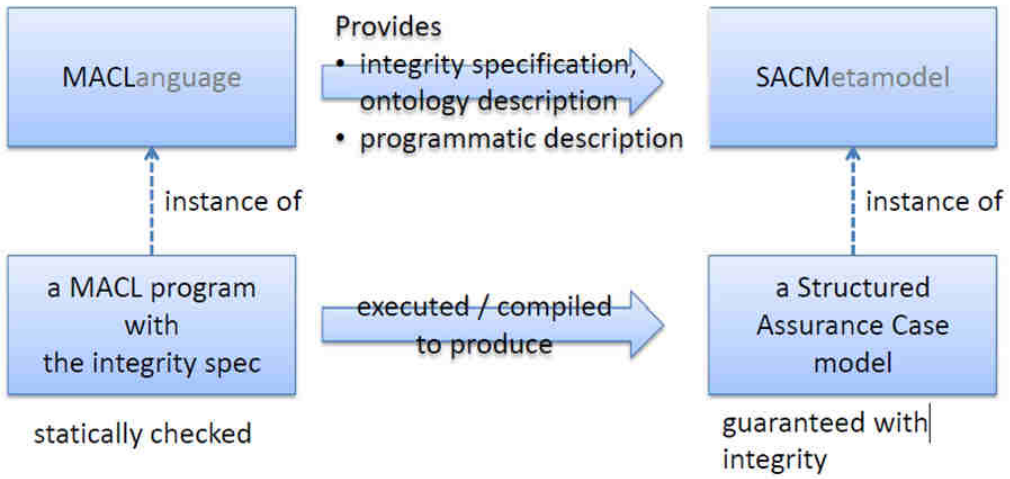


Fig. 73 MACL relation with SACM metamodels

Our proposal is aligned with the MACL vision where we apply the SACM extension metamodel for argumentation modelling and structured expressions are used in claims.

6.4 Formalized assurance contracts

As we have mentioned previously a contract can be defined as a collection of all dependencies between all subprojects of a project. In Fig. 74 it is shown the relationships between the components assurance subprojects that have been described in chapter 4, with the integrator assurance subproject, described in chapter 5.

Inside each of the subprojects we can identify the contract information that will include:

- Compliance maps that represents how the components is compliant to the standard requirements
- References to compliance elements from other subprojects

- Processes (activities) done by the components in relation with the integration as well as activities done in order to make reuse feasible.
- Evidence that is public and accessible to other subprojects.
- Public claims that can be reference by other subprojects but are justified inside the component
- Assumptions in form of claims or evidence that are reference inside the subproject but are supported in other subprojects.
- Pre-conditions, that are related to requirements for the component to make the reuse feasible
- Post-conditions which are resulting from the integration, these could include assertions in form of new requirements, or side condition or new activities.

All these aspects are validated in assurance contracts. The formalization of the assurance contract is focused on the public claims and assumptions included on the contract data from each of the components to be integrated.

In Fig. 75 the processes related with the assurance contract are shown. The assurance contracts have an impact in the activities performed by the component safety manager, the integrator and the system safety manager.

System safety manager will be in charge of system decomposition into components and define the system components architecture that is provided to the integrator. In many cases the system safety manager and the integrator safety manager are roles shared by the same person. Here we will treat the two roles separately.

The component safety manager is responsible for specifying the component data for the contract assurance. First, the component safety manager specifies assumptions about the context, the environment and use of the components. The specification of the assumptions are refined during the component development, and with each new refinement the level of technical detail increases. When the component is developed the guarantees should be included. Guarantees include the functional and performance behaviour of the component.

The component's assumptions and guarantees are included in the component argumentation model. Those claims that are considered guarantees shall be specified with the property public as true while the claims that are considered assumptions should be specified with the property assumed as true. We also considered component assumptions those information elements with the property type indicating context.

The integrator safety manager gets the system decomposition into components and the system architecture retrieve the information of the components from the components assurance subprojects. It is especially important the component's argumentation models as we have just mentioned, the component's assumptions and guarantees are specified there.

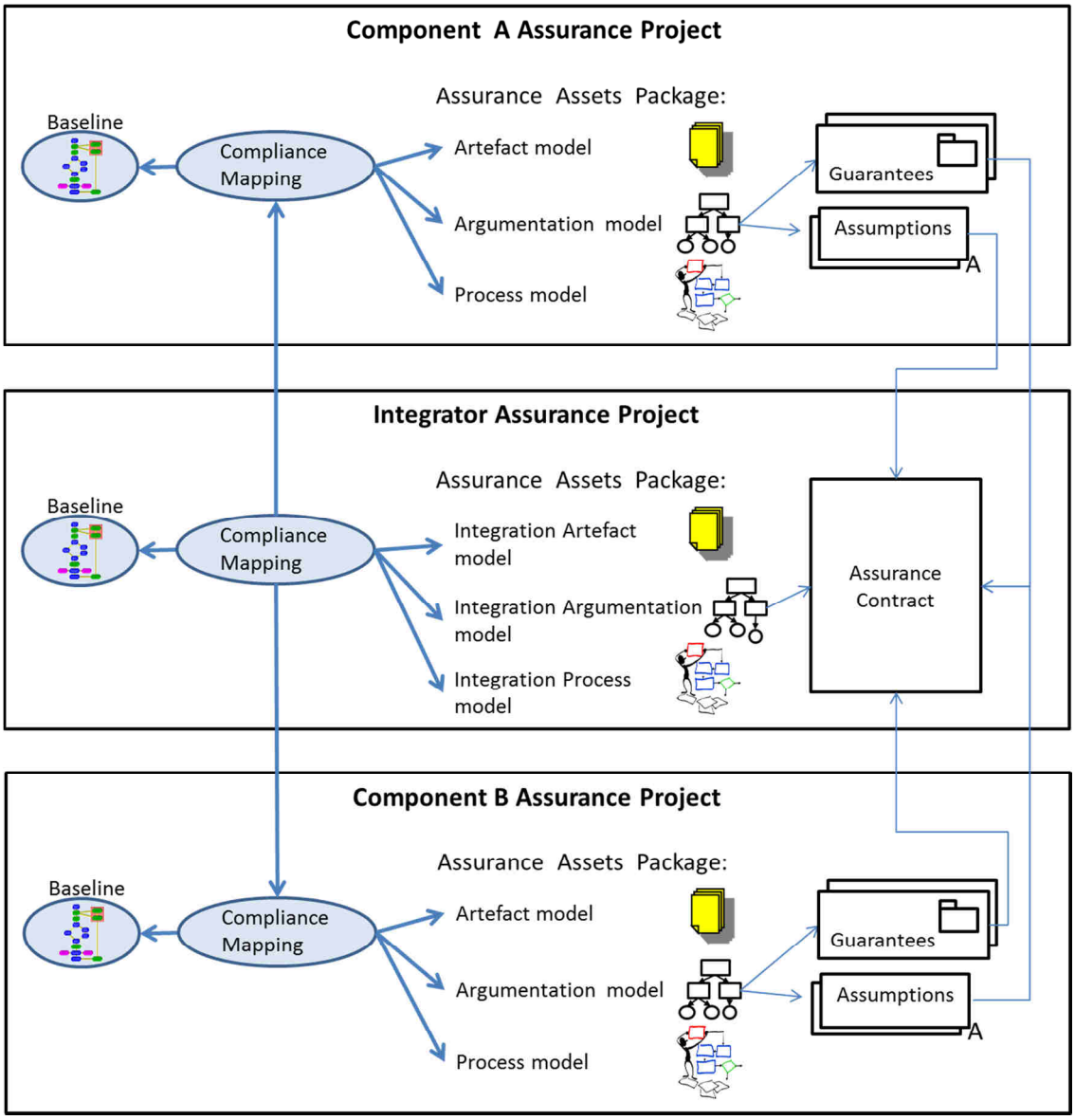


Fig. 74 Assurance subprojects relationships with assurance contracts

The integrator should take care of the reuse feasibility, the outcomes of this analysis is included in the integration argumentation model. The argument pattern for the reuse feasibility has been described in chapter 5.

After checking the reuse feasibility the component's assumptions should be validated to be included in the assurance contract. The integrator should also execute the post-conditions before the assurance contract can be validated and provide the system safety manager the integrated system.

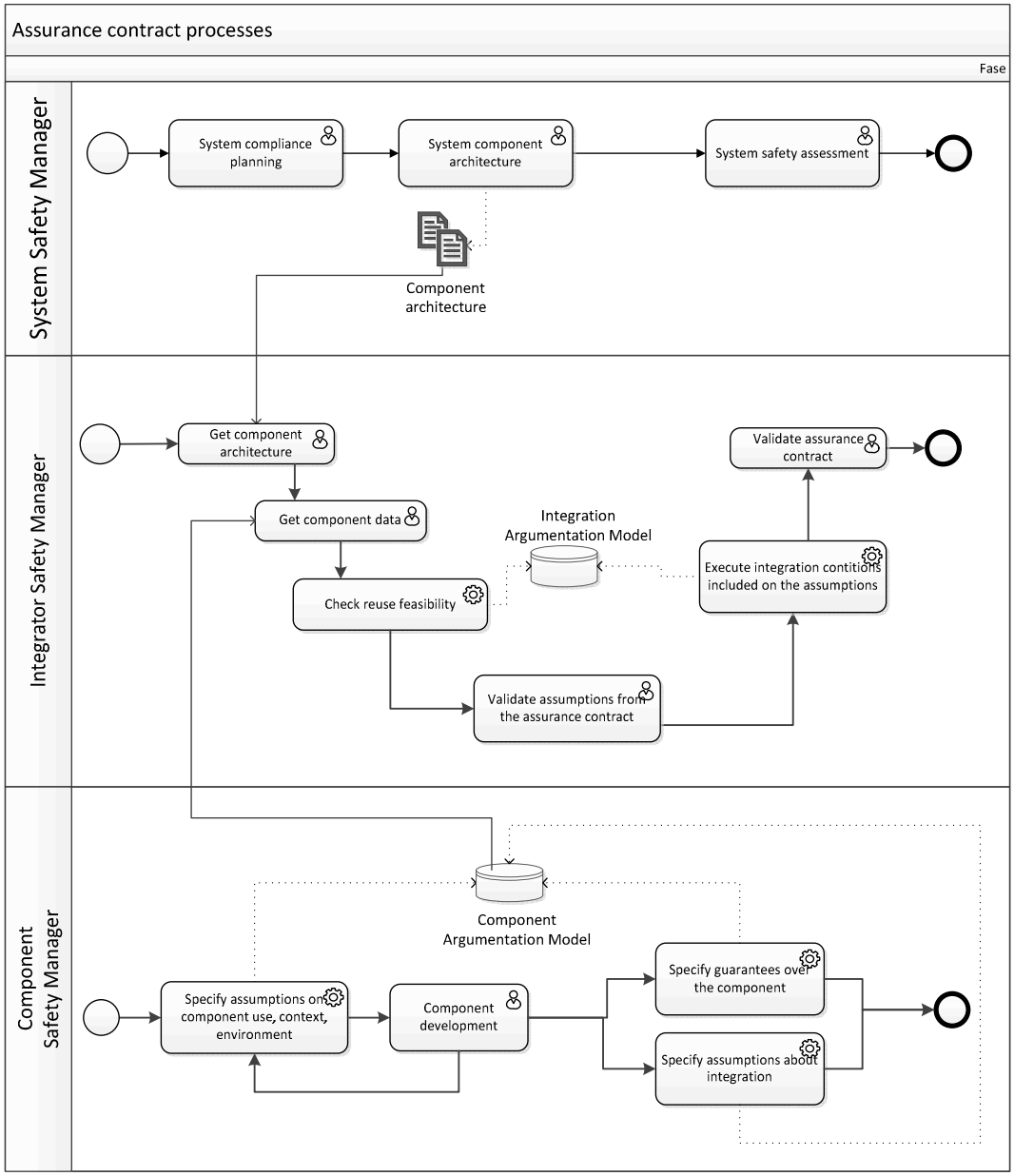


Fig. 75 Assurance Contract related Processes

The scenario of a simple assurance contract is when a component integrates within the system. In this case we will contract information data from the system and from the component. Each of the subprojects will have the “contract data”, this is the information from the component/system available to other subprojects and the collection of assumptions made during the development. This information should be available before the integration takes place. The contract will serve as the mechanism to compile in form of assertions the link between the assumptions and guarantees and should be evaluated as true to be valid. We also need a rationale when the link is not direct link. As it has

already been mention, the integration subproject will include an argumentation focusing on:

- The integration has correctly being done and all the pre-conditions and post-conditions have been fully accomplished.
- The links between the assumptions and guarantees are correctly justified and supported.

Assurance contracts can differentiate between two levels of specification. A first level or general level, depending on the aspect of the contract we are able to define a pattern. Also a company could include special requirements for contracts, for example an audition or specific analysis or a confirmation that there are not IP related issues for reuse.

Second level: at instance or project level. At this level, we need to do the specification of the contract data for an already defined component.

For the component is only possible to have post conditions at all if those are based on pre-conditions where that context is ever defined

Pre-conditions are inputs that will be artefacts; they provide knowledge about the usage scenarios. Those pre conditions should be defined before the integration time.

Defined assumptions are mapped to pre-conditions. Validated assumptions are mapped as post conditions.

One of the benefits of formalizing safety contracts will be the possibly of tool support for checking or generating contracts. Moreover, with the provision of a defined grammar for safety contracts we will be able to support validation of contracts (e.g. helping identify incomplete contracts). When defining this grammar, four characteristic where requested for the language to be specify:

- semi-formal
- finite
- exhaustive
- extensible

The contract language should be **formal and structured** in order to reduce ambiguity that has been detected as one of the challenges for assurance. With the proposal defined here, we will take advantage of the SACM extension meta-model presented on chapter 4 and extended it in order to benefit from concepts from ontologies approaches, and Semantics of Business Vocabulary and Business Rules (SBVR) and Backus Normal Form or Backus–Naur Form (BNF) rules so as to define the basis for formal and detailed natural language declarative description of an assurance assertion.

When we define our language as **finite**, we mean that the content of an assurance contract could include a finite number of concepts: processes, evidences and properties. These concepts in fact are the ones extracted from the analysis of the standard which has been described on chapter 3. Only these concepts will be included in the assurance contracts, other type of information is out of the scope of this language.

When creating an assurance contract we need to create assertions with sufficient expressiveness, for doing so with such a language, this should be **exhaustive**. We provide capabilities to deal with the concepts mentioned before. Nonetheless, the classification of the properties will serve as a guideline to check all the needed information is taken into account.

To deal with expressions used on assumptions and guarantees which are not covered we propose the use of either the informal **free-text** or either **extends** the list of structured expressions and links them to an existing category. We recommend that the contract creator use informal free –text the first time this time occurs and monitor the need. If the lacks continues with a certain frequency the best option is the extending the list of structured language expressions stored.

Guidelines from the standards offer the best practices and interpretations of the standards in order to comply with certain requirements. Those best practices can be modelled within the different technical approaches and impact on the methodology for the system development. Different technical measures can be put into place in order to assure the correct and complete following of the guidance and practices.

The use of contracts in component-based development is a well-known approach in the development of complex systems. It is based on the idea of “divide and conquer” where a complex development is done by the sum of various smaller and more manageable blocks or components that together form the whole. Joining blocks is a difficult task where the pointing and sharp borders need to life together without affecting one to another. On this environment is where contracts come out.

Contracts structure

The contract is structured into 4 main parts depicts in Fig. 76:

Definition
Assumptions - Activities/processes that shall be done by the integrator of the component - Properties of the component that shall be checked after the integration - Artefacts that shall be completed or done after the integration of the
Guarantees - Activities/processes that shall be done by the developer of the component - Properties of the component that shall be checked after the integration - Artefacts that should be completed or done after the integration of the
Rationale - Impacts on the guarantees if any of the assumptions is not valid - Correct reuse and integration - Rationale about the limits, conditions, and use of the component

Fig. 76 Contract structure

For the definition of the agreement two things must be taken into account:

- a unique identifier for the agreement in order to distinguish it from other contract that might be for the same system and
- a unique identifier of each of the subprojects involved on the agreement due to the integration.

The assumption part includes the collection of the assumptions made at the development time. The guarantee parts includes the public information about the subprojects involved that is accessible by other subprojects even though they are not involved at the moment on the contract. This part prepares the way to future contracts when new components are integrated on the resulting system from the first integration.

We should also include a rationale for assumptions validation as well as an argumentation about the correct reuse and integration.

6.4.1 Connections with the CCL

The parts of the contract are directly linked to the argumentation meta-model presented on chapter 4. The contract is identified by the agreement concept on the extended SACM meta-model proposed before.

This agreements or contract are done between argumentation that represents safety cases module for a specific component. In Fig. 77 the connection between the assurance contract and the CCL concepts are shown.

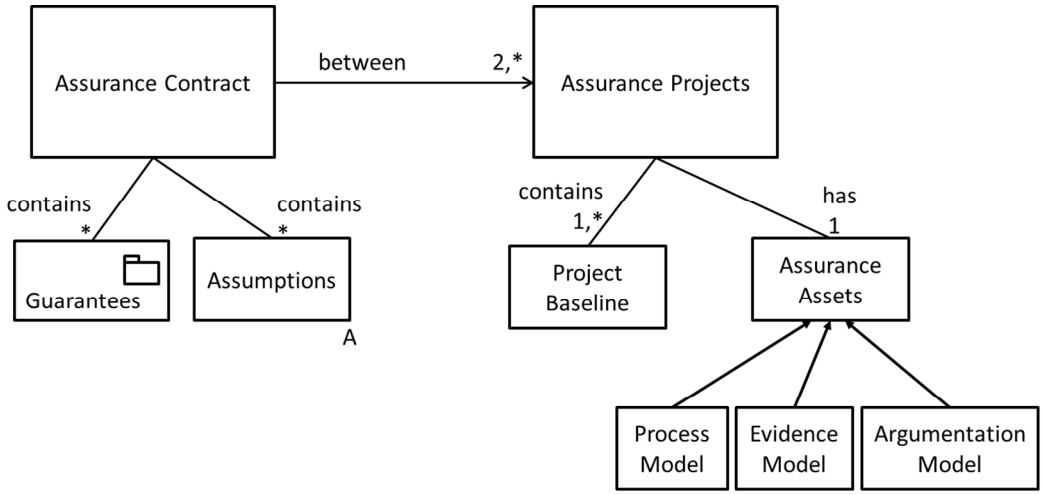


Fig. 77 Assurance Contract Connections with CCL

The assurance contract is a type of agreement. The assurance contract is done between at least two different assurance projects.

The contract consists of assumptions on the baseline referring to compliance maps in the component assurance project that will reference to an activity or an artefact. We can have a subtype of assumptions which are the pre-conditions and post-conditions which are described in a form of an assertion. They can be an assumed claim or an information element with property type as context.

As shown in Fig. 78, assumptions can be done about activities, properties and evidences. The activities that should be done after the integration can be seen as post-conditions of the reuse feasibility. Those activities should be included in the integration baseline and their process be monitored in the integration process model. Properties assumed represent the component expected context. Artefacts assumed should be included in the integration evidence model. Those artefacts should be released as they are requested by the standards and/or guidelines for compliance.

These claims could be about a property, evidence or a compliance element.

Guarantees will also be shown as assertion in the form of a claim. The difference from the assumptions because they are marked as public and other assurance project's elements are able to reference them.

Guarantees should be done about component properties. Guarantees about artefacts and processes do not necessary are included in the argumentation model as they are already reference by the compliance maps in the component assurance project.

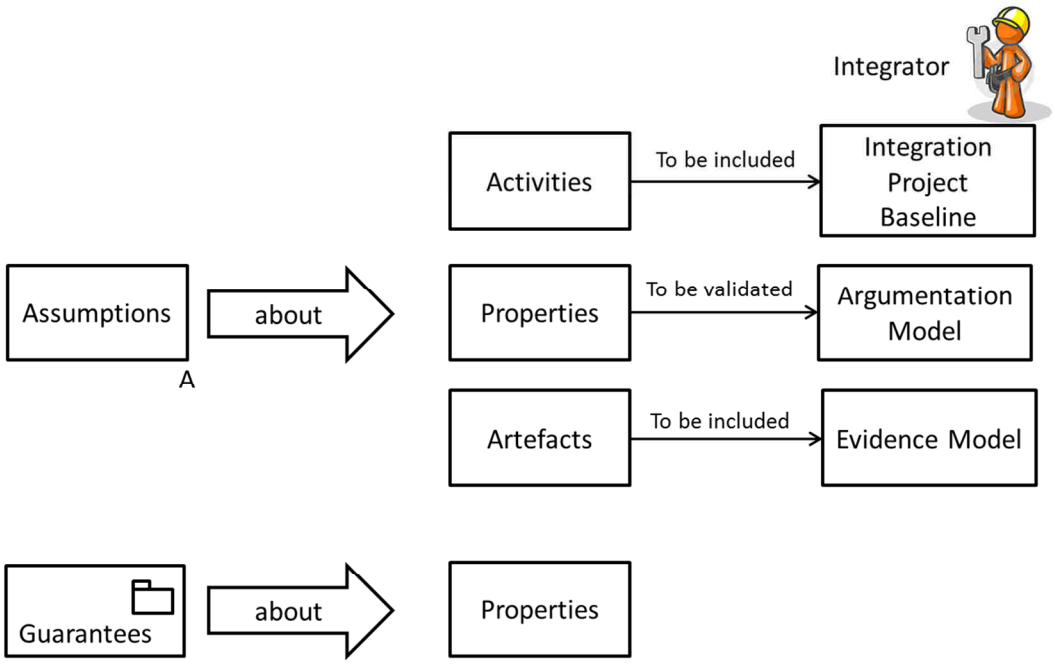


Fig. 78 Information contained in the assumptions and guarantees

Both, assumptions and guarantees can be extracted from the argumentation model. In the integration assurance project, we should include an argumentation model that will extract the information from the arguments model from the assurance subprojects involved. In this case, we will have a collection of all the claims set as public which are mapped as guarantees, claims set as assumed and information elements citations which are mapped as assumptions. The objective of this new argumentation is two folds, to link these assumptions and guarantees and to provide a justification about the correct reuse.

6.4.2 Structured expressions

The argumentation model for the assurance contract mentioned before is formed by assertions in form of claims. These assertions are usually made using natural language. The use of natural language can be counter-productive as it can induce to ambiguity which is one of the identified challenges we try to achieve with this proposal. The use of a formal language has a big barrier to be used on the industry to the difficulties to learn a new language. For that reason the use of structured expression formulated in natural language can be the solution.

We can leverage full benefit of the use of these controlled expressions by providing guidelines to the developers in accordance with the best practices of the guidelines for assurance and will be used by the people involved on the each of the component assurance. In compositional assurance it is important the use of coherent and adequacy terminology as we will need to "match up" information from different sources (assurance projects) created by different people and roles. This is particularly important in compositional argument structures, where the scope of the rely-guarantee agreements between components and services must be as clear as possible.

One important means of constraining the scope of the assertions is to specify the type and structure of claim structures which can be used. Claim types can be represented using structured expressions, where the nouns which can be used as grammatical subjects and objects are variable within defined limits, and where the verb phrases which carry the proposition (by governing the grammatical interaction of the nouns) does not vary.

In [Opencos D5.3] there is an initial list of claim types that we present in Table 11

Table 11 Initial claim types list.

Claim Type	Definition
Activity-Artefact Claim	Claims relating to the production of particular artefacts as a result of particular activities
Artefact Compliance Claims	Claims relating to the necessity of particular artefacts for compliance.
Artefact Adequacy Claims	Claims relating to the adequacy and appropriateness of particular artefacts – i.e. moving beyond compliance into a justification of the evidence artefacts provided. E.g. the adequacy of a fault tree
Activity Compliance Claims	Claims relating to the necessity of particular activities for compliance
Activity Adequacy Claims	Claims relating to the adequacy and appropriateness of particular activities – e.g. the suitability of a particular analysis technique.
Component Development Claims	Claims relating to the adequacy of the process by which a iomponent has been developed

Claim Type	Definition
Fault Accommodation Claims	Claims relating to the accommodation or elimination of a fault
Hazard Mitigation Claims	Claims relating to the adequacy of hazard mitigation achieved by safety measures in the design

Claim types for this table have been identified for different sources:

- An analysis of existing GSN argument patterns [Hawkins Kelly 2013].
- The Safety Case Repository assembled by the Dependability Research Group at the University of Virginia [Virginia].

In order to enrich this list we have taken into account the analysis of the standard presented on chapter 3. The information about the processes and artefacts are covered by the claim types identified, however, not all the properties identified are covered. Initially we have categorised the properties into: Installation, Safety, Functionality, Performance and Timing according to the different concerns a reusable software component should pay attention as it is mentioned on AC 20-148.

We have focused here on the safety and performance categories and make a comparison across the different standard and extract some new types of claims.

Table 12 Claim types list based on standard's analysis.

Claim Type	Definition
Agent Action Claims	Claims relating to the role of agents (external to the system), particularly where this affects system safety. Note that agents may be non-human (i.e. these are environmental claims)
Environmental property claims	Claims relating to conditions in the operating environment which have a bearing on the safety of the system
Overall Safety Claim	High-level claim about the acceptability of the safety of an entity, which is usually the root node in an argument
Element Performance/ behaviour claim	Claims relating to the functionality and behaviour of some system elements
Element adequacy claims	Claims relating to the appropriateness of certain characteristics of elements (e.g. weight, timing)
Element compliance claims	Claims related to the level of compliance of an element to a specific standard

Extending the CCL in [OPENCOSS D5.6] a Metamodel for Structured Expressions is proposed as an extension to the Structured Assurance Case Metamodel [SACM 1.1].

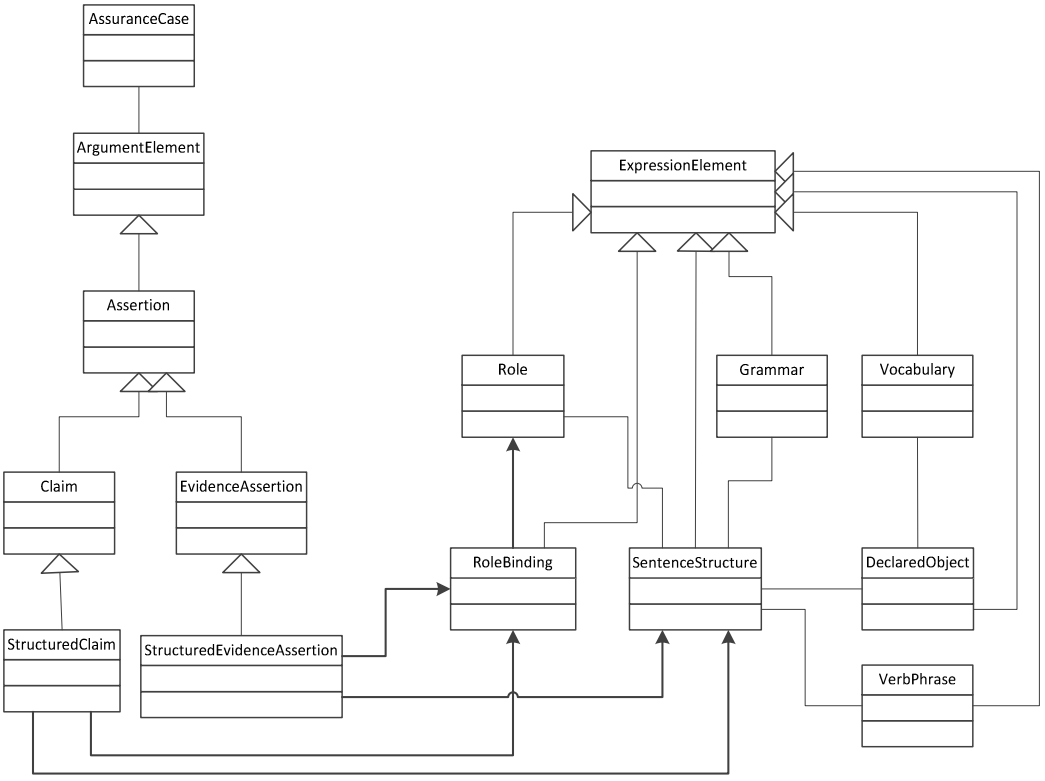


Fig. 79 Structured Expressions Metamodel.

The next step has been the identification of structures expression and map them to the categories already identified. For doing so, we do not only look at the sources used for defining the claim types but also information provided as inputs for the execution of the case studies that will be presented on following chapters.

Table 13 shows the connection between the claim types and the identified structured expressions. For the structured expression definition we used (Backus Normal Form or Backus–Naur Form) grammar.

Table 13 Structured expressions proposed by each claim type.

Claim Type	Structured Expression Example
Overall Safety Claim	{item|platform|system|subsystem|element|component} is acceptably safe to operate in its defined context
Activity-Artefact Claim	1. {artefact} was produced during {activity} 2. {artefact} was produced during {activity} using {technique} 3. {artefact} was produced using {technique}

Claim Type	Structured Expression Example
Artefact Compliance Claims	1.{artefact} satisfies {requirement} 2. {artefact} satisfies {requirement} of {safety standard|guidance document} 3. {artefact} has been generated in accordance with the requirements of {safety standard|guidance document} 4. {artefact} has been generated using {technique} in accordance with the requirements of {safety standard|guidance document} 5. {artefact} has suffered {action}
Artefact Adequacy Claims	1. {artefact} has {artefact property} 1a. {artefact} is sufficiently {artefact property} 1b. {artefact} shows {artefact property] to an appropriate degree 2. {artefact} provides adequate support for {objective}
Activity Compliance Claims	1. {technique} has been applied in accordance with the requirements of {safety standard|guidance document} 2. {technique} satisfies {requirement} 3. {activity} has been carried out in accordance with the requirements of {safety standard|guidance document} 4. {activity} satisfies {requirement}
Activity Adequacy Claims	1. {technique} used for {objective} has {technique property} 1a. {technique} has {technique property} 1b. {technique} is sufficiently {technique property} 2. {technique} was carried out by {person|role} 3. {activity} has {activity property} 3a. {activity} is sufficiently {activity property} 4. {activity} satisfies {objective} 5. {activity} used for {objective} has {activity property}
Component Development Claims	1. {development technique} is appropriate for the development of {item|platform|system|subsystem|element|component} 2. {development technique} has an appropriate level of {development technique property} 3. {development technique} is sufficient to meet {requirement} 4 {Parameter } is defined by {identification} with {range}

Claim Type	Structured Expression Example
Fault Accommodation Claims	1. {fault|failure} is diagnosed by {item|system|element|component|agent} 2. item|system|element|component|agent} diagnoses {fault|failure} 3. {fault|failure} is diagnosed by {item|system|element|component|agent} and indicated by {action} 4. {fault|failure} is indicated by {condition} and detected by {item|system|element|component|agent} 5. {fault|failure} is adequately mitigated by {fault accommodation technique} 6. {fault|failure} is adequately mitigated by {agent behaviour} 7. {fault|failure} is adequately mitigated by {system|element|component property}
Hazard Mitigation Claims	1. {hazard} has been identified 2. {hazard} is adequately mitigated by {hazard mitigation technique} 3. {hazard} is addressed by {design technique} 4. {hazard} is adequately mitigated by system|element|component property}
Agent Action Claims	1. {agent} will {action}{condition} 1a. {agent} will {action}{condition} when {stimulus} 2. {traffic-participant} will {action}{condition}
Environmental property claims	1. {environmental property} is fixed 2. {environment agent} will {action} when {condition|stimulus} 3. {environment property} is <value> 4. the rate of {environment property} is <value>

Claim Type	Structured Expression Example
Element behaviour claims	1. {item|system|subsystem|element|component} performs {function} 2. {item|subsystem|element|component} performs {function} with {fuction property} <value> 3. {item|system|subsystem|element|component} will perform {function} when {condition|stimulus} 4. {item|system|subsystem|element|component} will perform {action|function} to maintain {state} when {condition|stimulus} 4b. {item|system|subsystem|element|component} will perform {action|function} to achieve {state} when {condition|stimulus} 4c {item|system|subsystem|element|component} will perform {action} to avoid {state} when {condition|stimulus} 4d. {item|system|subsystem|element|component} will perform {action} to prevent {state} when {condition} 5. {state} will be prevented when {condition} 5a. {state} will be achieved when {condition} 5b. {state} will be avoided when {condition} 5c. {state} will be maintained when {condition}

Claim Type	Structured Expression Example
Element adequacy claims	1. {item|system|subsystem|element|component} has {item|system|subsystem|element|component property} 1a. {item|system|subsystem|element|component} has {item|system|subsystem|element|component property} <value> 2. {item|system|subsystem|element|component} as an appropriate degree of {item|system|subsystem|element|component property} 3. {item|system|subsystem|element|component}'s {item|system|subsystem|element|component property} is acceptable 4. {item|system|subsystem|element|component}'s {item|system|subsystem|element|component property} satisfies {requirement} 5. {item|system|subsystem|element|component}'s {item|system|subsystem|element|component property} is <value> 6. {item|system|subsystem|element|component}'s {item|system|subsystem|element|component property} is sufficient 6a. {item|system|subsystem|element|component} {item|system|subsystem|element|component property} is sufficient to show {objective} {item|system|subsystem|element|component}'s {item|system|subsystem|element|component property} is <value> determined by {technique}
Element compliance claims	1 {item|system|subsystem|element|component} shows {'partially compliance | full compliance | not compliance} a level of compliance to objective {objective id} for the standard {standard} to level {criticality level} 2 {Objective| requirements} specific mean of compliance is {Verifiable| Formal Review/Audit | Analysis | Test| Inspections | Process Evaluation |Software Certification Data } and the status is {complete, approved and released | still in progress}

6.5 Contracts lifecycle

The contract specification progresses through the development lifecycle. Similar to the development, the contract has its own lifecycle as well. In following Fig. 80 we try to express the different phases of the contract.

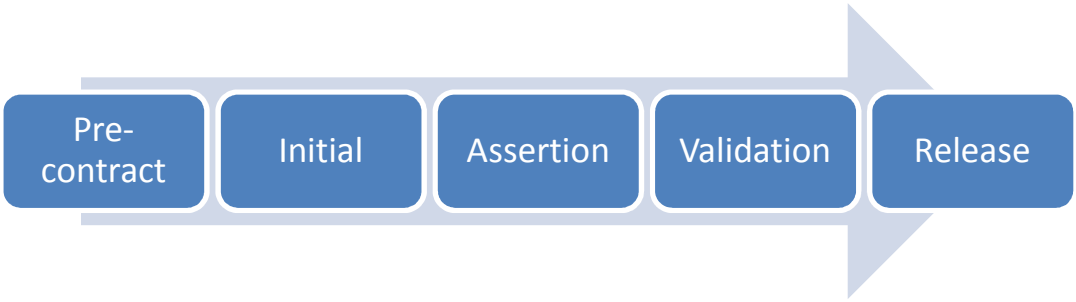


Fig. 80 Contracts Development Phases

Pre-contract phase — During this phase we will need to ensure that general requirements for reuse have been fulfilled. At this stage we need to check whether the company defined requirements for reuse (for example, IPR analysis) has been done and the results permit the reuse. As a result of this stage we will be able to have a template of a contract created, where the first level of abstraction of the contract is specified.

Initial phase — From this stage on the contract is project-specific. In this phase we will reference the different assurance interfaces and recompile the contract data from each component.

At development time we should extract from the project the different assumptions made and the guarantees. It is important at that phase that we take advantage to use structured expressions mentioned before and used them on the contract interfaces. In the argumentation models for the components the public claims will be considered the guarantees and the assumed claims and the information element citations will be considered the assumptions of the components. This is the last phase done at component/assurance project standalone level.

Once the component is going to be reused, we should argue about the feasibility of the component reuse. In order to follow the best practices and have a complete argumentation, we have created an argumentation pattern that we explained in chapter 5 (see Fig. 62).

Assertion phase — In this phase we need to map the interfaces in order to create non-ambiguous assertions. We also need to declare post conditions to be met once the agreement is done and to define the rationale behind the assertions. In the integration assurance project, we will create a new contract inheriting all the assumptions and guarantees from the assurance projects involved. We will also include a new argumentation model focusing on the correct integration and the feasibility of the reuse.

Validation phase — In this phase, each of the assertions defined during the previous phase needs to be evaluated as either true or false. If any of the assertions is declared false as a result of the evaluation then all of the agreement must be considered to be invalid and further analysis needs to be undertaken. All the assumptions and guarantees need to be linked and trace, for doing so, the classification based on the structured

expression used will be beneficial as we link assumptions and guarantees of the same typology.

Activities must be checked first because they produce artefacts. Evidence artefacts could be used to support part of the assurance case, and therefore need to be valid in the context of use.

Release phase — In this phase, all of the post conditions that were declared during the assertion phase need to be executed and validated. Only after those conditions are executed can the agreement be considered to be valid

Premises and promises are the core of the contracts. Premises need to be validated before the contract promises can be fulfilled. Those premises are typically identified at the component level. Promises can be made at component level but also new promises can appear as the integration of components enables new promises (regarding the composition of components) to be made. The documentation of assumptions and intended context of use are seen as premises here. They indicate the boundaries and operation conditions that ensure the correct and safe use of the component.

Promises and premises are closely interconnected. Guarantees identified at component level but promises that are not ensured and validated by contracts could make the contracts not valid. It is also important to consider behaviour, not only nominal behaviour but also failure and degraded behaviour are important to consider for both the safety contracts and assurance contracts.

6.6 Conclusions

In Table 14 we have tried to summarize the issues and challenges which were identified in 6.2 Challenges section and how we have tried to give them a response.

Table 14 Conclusions to challenges identified in chapter 6

Id	Challenge	Challenge Description	Mechanism	Solution description
1	Human factor	People get tired, distracted, and can they can omit some details. System complexity is being managed by decomposition into components however, the assurance should be treat at system level and this complexity could affects those in charge of assuring the system.	Categorization	The claims typology and the data base with the possible strutted expression offer the user the support needed to take into account all the views which are needed on the contract creation and validation

Id	Challenge	Challenge Description	Mechanism	Solution description
2	Validation and checking	The possibility of tool support for checking or generating contracts	Formalized contracts	Having a contract structure and the structured expressions provide the basis for the tool support for creating the contracts
3	Interoperability between different suppliers	We also need to express the guarantees the component offers on a uniform way to the different people which participate on the system development, so all of them know what is expected from them and what can they ask to that particular component.	Unique language	Having a unique and understandable language to express what is a guarantee and an assumption. The language distinguish the parts of the contract and reduces the ambiguity
4	Facilitate the integration of the components within the system	Having a structure and clear way to define the interfaces of the component from the assurance perspective will benefit the integration. At integration phase we have seen the contract based approach for technical integration for some properties such as timing	Categorization Reuse pattern	The claims categories offer a good option to check the complexity of the integration. The reuse pattern also provides support to ensure the correct reuse of a component.
5	Learning curve	The learning curve to learn a formal language is big barrier for the introduction of those languages on the industry	Structured expression	Using structured expressions formalized in a natural language

"What is now proven was once only imagined." Hindsight is a wonderful thing but foresight is better, especially when it comes to saving life, or some pain! – William Blake

7

Case Studies

7.1 Background

According to [Yin 2013] a case study is the preferred method when '(a) "how" or "why" questions are being posed, (b) the investigator has little control over events, and (c) the focus is on a contemporary phenomenon within a real-life context'.

Runeson and Höst [Runeson Höst 2009] indicate that "The analytical research paradigm is not sufficient for investigating complex real life issues, involving humans and their interactions with technology". Runeson defines the study objects for a case study in the scope of software engineering as those corporations or agencies that develop software, which are project oriented with an advanced engineering work behind.

This chapter presents the industrial case of the study used to benchmark the thesis objectives. The study objects are private corporations: Thales, France for the avionics case study, Centro di Ricerche di Fiat, Italy for the automotive case study and NUTES, Brazil for the case related to the medical device study. In all the cases the study is focused on a given industrial project, the corporations are working on and in all the cases the people involved where either Ph.D. or long experienced (more than 12 years) engineers.

Case studies are often criticised aspects of the case study for being of less value and impossible to generalized from. In order to cope with this issue, the idea has been to provide three case studies in three different industrial domains using the same template in order to harmonize the way their specifications are presented.

An evaluation framework has also been developed and all case studies will be evaluated using the same procedure. The evaluation framework will be described and the results explained in the next chapter. This chapter is focused on the design of the case studies and how they have been implemented using the approach presented in previous chapters.

The objective when defining the following case studies was to answer the question on how should we manage compositional assurance data in order to:

- Reuse previous developments
- Reduce certification costs

In order to verify the feasibility of the approaches presented in this thesis, all the case studies will follow the same approach and apply the same methodology (See Fig. 2). The case study development cycle consists of 5 process steps: (1) design the case study by preparing the procedures, (2) prepare to conduct the case study pilot, (3) collect the evidences for the case study, (4) analyse the case study evidence in contrast with the proposed theory and (5) share the case study report. This chapter describes the work done in steps 1, 2 and 3. Chapter 8 will focus on steps 4 and 5.

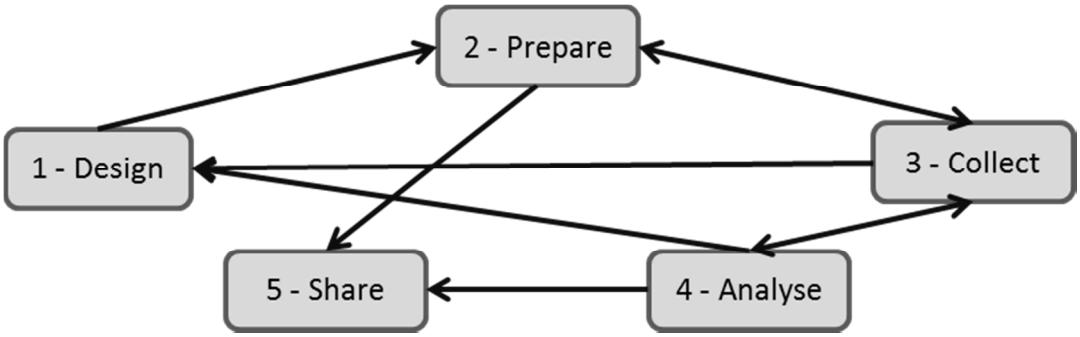


Fig. 81 Phase 2 of the thesis methodology, case studies development cycle

The approach has been developed inside the OPENCOSS project, presented to all partners of the project in October 2014 and to NUTES team in March 2015.

Each case study has been described for the design following this template.

1. **System Description**
   [Presentation of involved actors, operational scenarios, architectural model of the system and main functions.]
   1.1. Industrial use case actors and environment
   [*Diagram describing the interfaces between the system and the actors*]
   1.2. Industrial use case operational scenarios
   1.3. Main functions provided by the system
   1.4. Architecture of the system
   1.5. General characteristics of the system
2. **Description of the Compositional Approach**
   [Description of which items in the industrial use case are re-used from other contexts. Identification of which components are pre-existing but undergo modification.]
3. **Summary of main argument for safety**
   [This is a capsule summary of the case for safety that is presented in the use case, to facilitate identification of the most important aspects.]

7.2 Avionics case study

7.2.1 System description

The avionics safety assessment process is depicted in Fig. 82. It includes the system development process because that is closely interlinked with the safety assessments; for example, the safety requirements are coming from safety standards and have a direct

influence on the product, while the evidence for demonstrating safety are based on the test results of the product. This makes it impossible to look at the safety assessment without considering the development process.

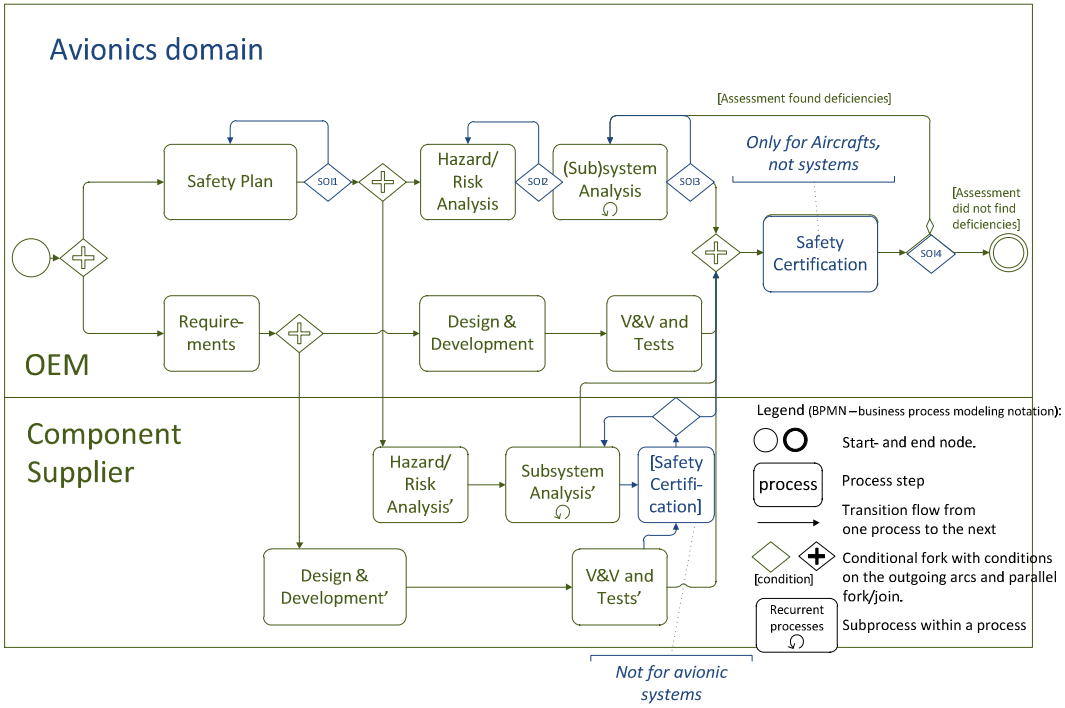


Fig. 82 Safety assessment business process for the avionics domain

Another important aspect is included in the overall safety assessment process: the process of the component supplier. As such it is important to know how the process of the component supplier is interwoven in the total system safety assessment process.

In avionics there are three levels of development and construction activities: the platform or aircraft level, the system level, and the item or component level. Platforms are created by aircraft or rotorcraft manufacturers, components by equipment or component providers. Certification only happens at these two levels: the platform or aircraft level and the physical component level. Avionic systems are not yet certified as standalone systems, even though progress is made in this direction with IMA (Integrated Modular Avionics) certifications.

The execution platform is considered as an independent item for which a qualification dossier will be built. This qualification dossier consists of plans, technical documents, and certification documents. Technical documents are specifications, validation and verification life cycle data. The certification documents are configuration index documents and accomplishment summaries.

7.2.1.1 Industrial use case study actors and environment

The use-case proposed is relative to IMA general environment. This case study is focused on IMA platform integration specially related to the partitioning services. Fig. 83 highlights the focus of the avionics case study in relation with the IMA elements: a software component from the platform (see grey boxes).

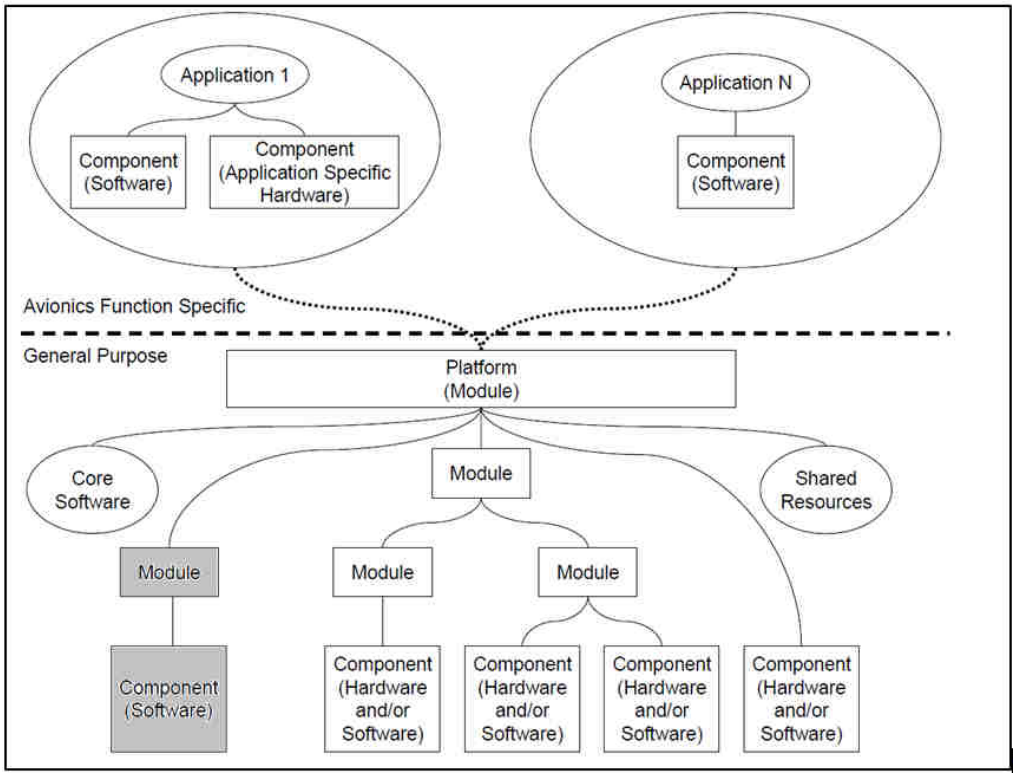


Fig. 83 Focus of the avionics case study in relation of the IMA elements

The Execution Platform is a common resource set used by several actors:

- It is defined and validated by the IMA Platform Architect
- It is provided by an Execution Platform Supplier, also called Module Supplier
- It is also used by Function Suppliers in charge of developing applications

Tools set is also delivered with the Execution Platform by the Module Supplier for the users (IMA Platform Architect and Function Suppliers), containing at minimum:

- Operative System configuration tools
- Operative System Simulation tools

Remark: the use-case will be focused on technical data to be provided for Hardware and Operative System Tools will not be considered.

The main technical interfaces to be managed for the Execution Platform are:

- The API defining all services offered to applications

- The characteristics of Execution Platform and constraints for using it, necessary to establish Usage Domain of the IMA Platform

The main relation relative to the use-case is between IMA Platform Architect and Module Supplier: the IMA Platform Architect will formally accept the Execution Platform provided by Module Supplier.

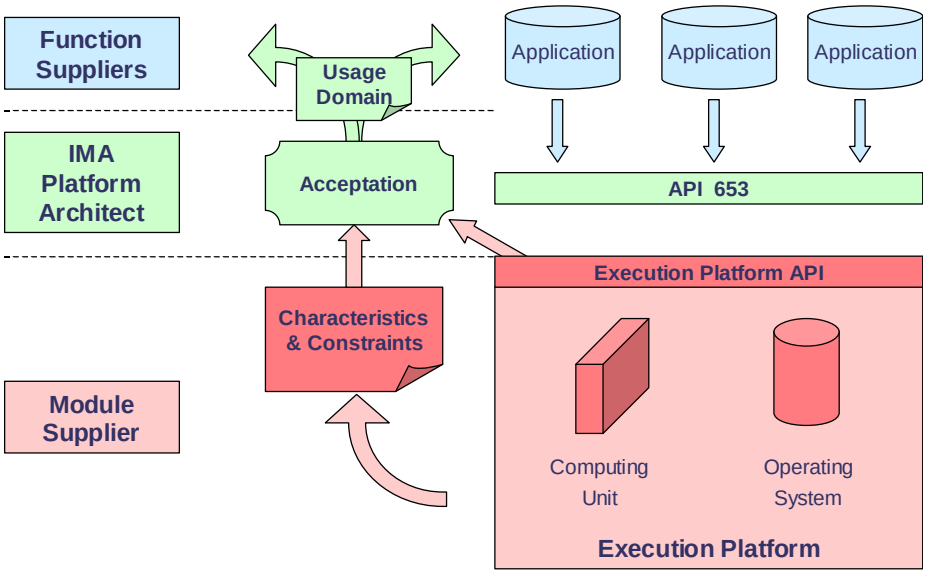


Fig. 84 Use-Case environment and actors

7.2.1.2 Industrial use case operational scenarios

Use-Case operational scenario is the following:

IMA Platform Architect establishes general hypothesis and certification baseline:

- Sizing hypothesis (memory, processor throughput)
- Certification standards applicable (DO-254, DO-178C …)
- Functionality expected (API A653 …)

IMA Platform Architect fixes Execution Platform perimeter for Module Supplier:

- Hardware (Processing, IO, Mass Memory …)
- Software (Operative System, drivers, Platform System functions …)

Module Supplier provides Usage Domain (characteristics and usage constraints)

Module Supplier provides qualification material for certification demonstrations

IMA Platform Architect validates Module Supplier data and provides formal acceptation

7.2.1.3 Main functions provided by the system

Main resources and services provided by the Execution Platform are:

- Computing resource (processor)
- Operating System including:

  - Operative System heart providing:
- capability to manage real time scheduling
- capability to manage application processes
  - Drivers offering capability to access to Inputs/Outputs and network

Additional services (Platform System Applications) present on IMA Platform are not in the perimeter of the Execution Platform:

- Initialization and modes management
- Data-loading capability
- Monitoring and BITE capability

7.2.1.4 Architecture of the system

Typical IMA Platform architecture is the following:

Hardware level composed of:

- CPU board supporting avionic network interface
- IO boards connected to CPU via PCI internal bus

Operative System level composed of:

- Operative System kernel
- Input/Output drivers addressing Input/Output boards

IMA System level composed of system applications handling platform level services such as Data-Loading, BITE, Instrumentation

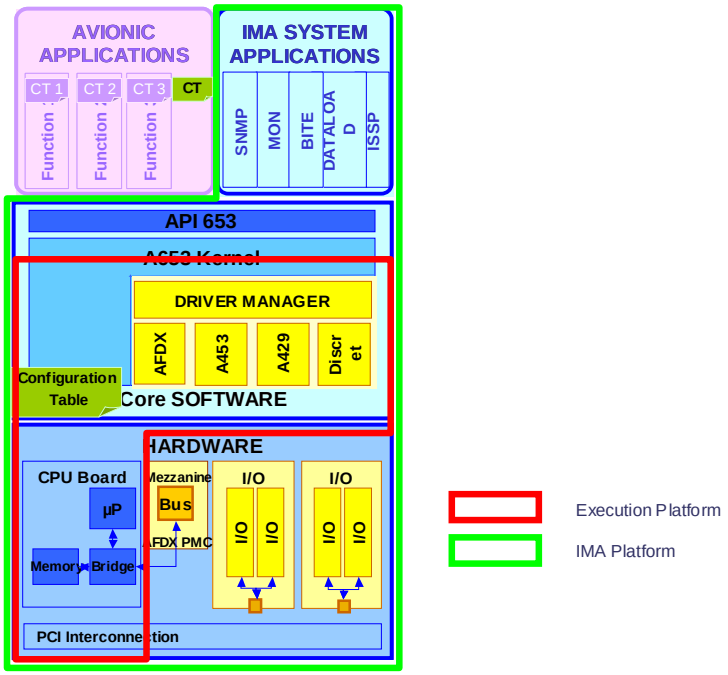


Fig. 85 Execution Platform & IMA Platform block diagram

Avionic Applications are installed on the IMA Platform and use offered resources via A653 API

7.2.1.5 General characteristics of the system

Environmental conditions: defined by DO-160 categories

Performances: capability to manage several avionic applications on one processing unit

Safety constraints:

- robust partitioning capable to support incremental certification process
- design insuring determinism: independence between parts of the Execution Platform (IO / IO, Network / IO, no demon processes in background, performances characterized in worst case, …)
- Undetected failure rate at 10^{-6} / h
- Loss rate at 10^{-5} / h

7.2.2 Description of the compositional approach

The overall context is presented in Fig. 86. The following elements are identified in an integrated platform. These elements should be considered separately in the compositional approach, because they are provided by various actors:

- The Execution Platform: object of the use-case, reused from another project,
- Boards of computing module: provided by other suppliers, should not influence or be influenced by Execution Platform definition, except through identified interfaces
- Platform Applications are applications implementing transverse services for the platform (Data-loading, BITE …): provided by other suppliers, should not influence or be influenced by Execution Platform definition, except through identified interfaces (API & Usage Domain)
- Common Configuration: includes configuration parameters expressing the way the resource is organized and distributed to various applications. Provided by Platform Integrator, should not influence or be influenced by Execution Platform definition, except through identified interfaces (Usage Domain)
- Applications: implement avionic functions. Provided by Function Suppliers, should not influence or be influenced by Execution Platform definition, except through identified interfaces (API & Usage Domain).

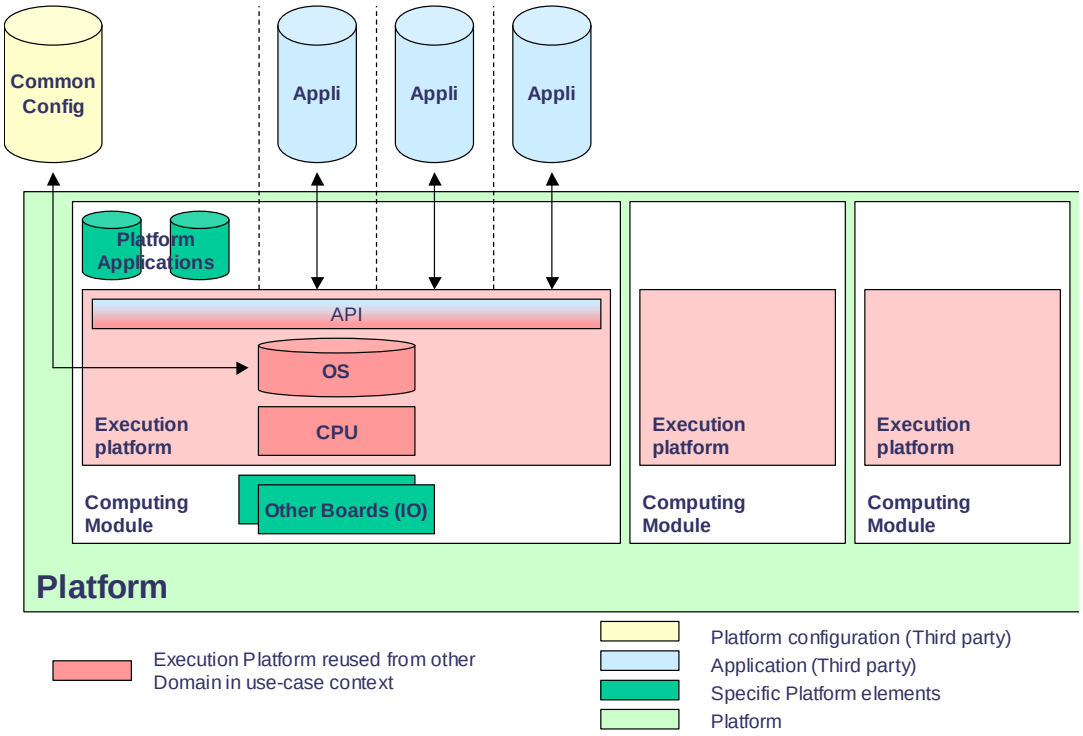


Fig. 86 Compositional approach for avionics industrial case study

In the context of this case study, only part of the execution platform will be considered. This also represents a potential real industrial case where only part of platform is reused from another project (typically Operative System heart).

As shown in Fig. 86, several items are composing the execution platform:

- Operative System kernel
- Drivers
- Hardware items (boards)

All items (software or hardware) should be considered as independent, provided interfaces are defined:

- Hardware / Software interface
- Driver / Operative System interface

The proposed used-case focuses on data to be exchanged between stakeholders, mainly hardware and/or software data.

7.2.3 Source data

Life cycle data used from the avionics domain are described in the Table hereafter (Table 15).

Those documents were mainly used to provide evidences while creating the DO-178C model and illustrating the typical artefacts used in an avionics project. Those artefacts answer to both DO-178C standard and internal company referential.

Table 15 Avionics Life Cycle Data

Document	Life Cycle Data provided
Checklist_HLR.xls	This document provides requirements for High Level Requirement Verification
Checklist_LLR.xls	This document provides requirements for Low Level Requirement Verification
Opencoss_PAS_Notebook.doc	This document provides the Process Accomplishment Summary
Opencoss_PRS.doc	This document provides the Process Requirement Specification
Opencoss_PSAC.doc	This document provides the Plan For Software Aspects Of Certification
Opencoss_SAS.doc	This document provides the Software Accomplishment Summary
Opencoss_SDVS.doc	This document provides the Software Development And Verification Standards including rules for Software Requirement Specification, High Level Requirement , Software Design, Low Level Requirement, Software Verification Procedure and Software Test Document
UD_Eamples.doc	This document provides the Usage Domain Requirements
Eurocae ED-12C	Software Considerations In Airborne Systems And Equipment Certification
Eurocae ED-124	INTEGRATED MODULAR AVIONICS (IMA) DEVELOPMENT GUIDANCE AND CERTIFICATION CONSIDERATIONS
SAE ARP 4754A	Guidelines for Development of Civil Aircraft and Systems

7.2.4 Case study implementation

7.2.4.1 Assurance Compliance Modelling

In this step the Standard expert will follow the process already described in chapter 4 which is shown in Fig. 87. Information Source used are ED-12/DO-178 and ED-124/DO-297.

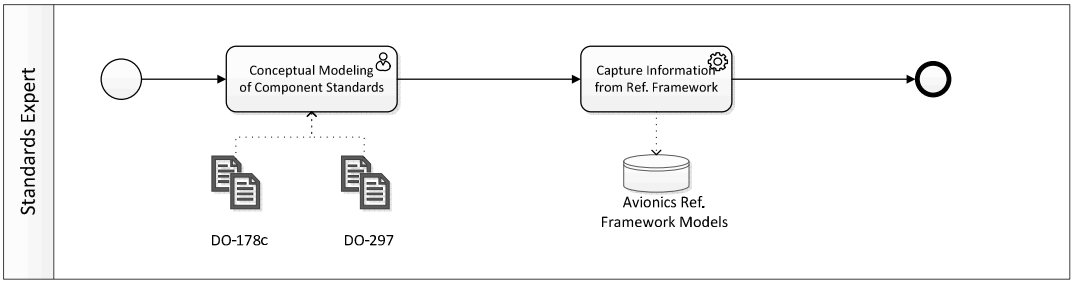


Fig. 87 Process followed in the avionics case study by the standards expert

We have modelled both standards into two different reference frameworks. In Table 16 one of the tables included as annexes on the DO-178C standard can be seen.

Table 16 Excerpt of table A.1 from DO 178c annexes [DO 178c].

Objective	Activity	Applicability by Software Level					Output	Control Category by Software level				
Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
The activities of the software life cycle processes are defined	4.1a	4.2a 4.2c 4.2.d 4.2e 4.2g 4.2i 4.2l 4.3c	O	O	O	O	PSAC SDP SVP SCM Plan SQA Plan	11.1 11.2 11.3 11.4 11.5	1 1 1 1 1	1 1 1 1 1	1 2 2 2 2	1 2 2 2 2

Objective	Activity	Applicability by Software Level					Output	Control Category by Software level				
Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
The software life cycle(s), including the inter-relationships between the processes, their sequencing, feedback mechanisms, and transition criteria is defined.	4.1b	4.2i 4.3b	○	○	○	○	PSAC SDP SVP SCM Plan SQA Plan	11.1 11.2 11.3 11.4 11.5	1 1 1 1 1	1 1 1 1 1	1 2 2 2 2	

The column Objective is modelled as the objective parameter inside a compliance activity. The columns activity Ref is modelled as requirements for the compliance activity. The applicability software levels are modelled using applicability tables. The output column is modelled as artefacts with an arrow going out from the activity and pointing to the artefact.

In Fig. 88 an excerpt of the DO 178c standard is shown.

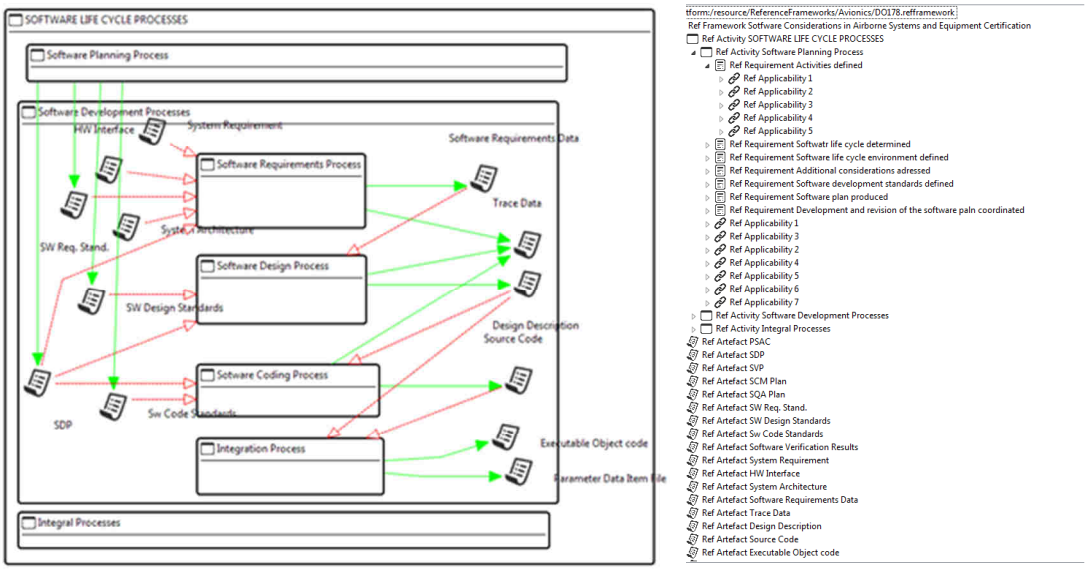


Fig. 88 Excerpt of DO-178c model

For the use case we have modelled the DO-178c standard but just some parts of the DO-297, just the part about IMA module/platform development process (Task 1) and part about module or application reuse (Task 6).

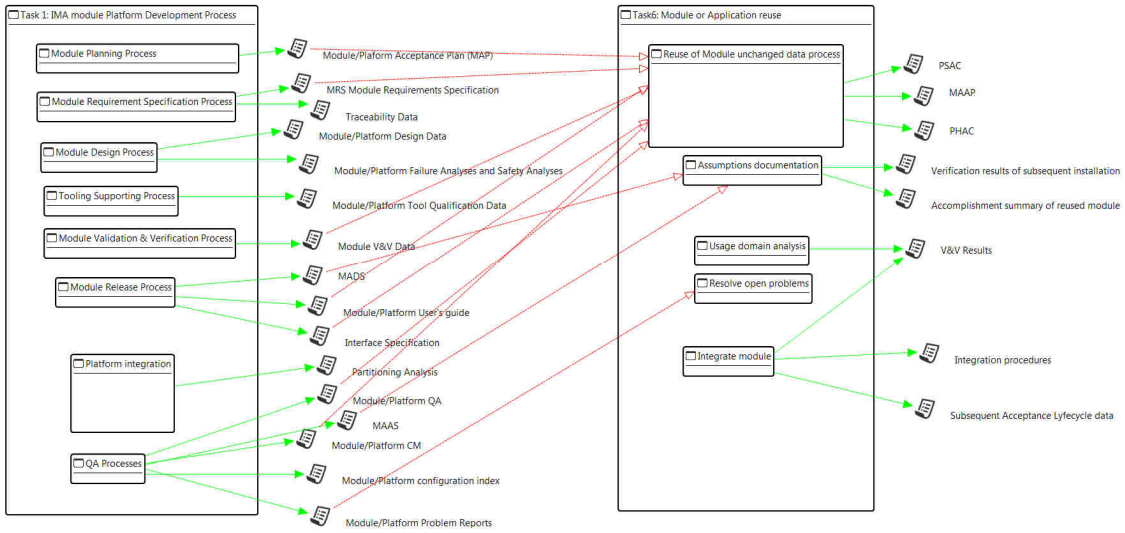


Fig. 89 Excerpt of DO-297 model

7.2.4.2 Assurance Modeling for Platform/Component

For the use case we have the "AvionicsMACSPlatform" assurance project which focuses on the development of the Execution platform and The Execution platform complies with the DO 178C and some parts of the DO-297. In the execution platform assurance project there are two baseline, one baseline per standard to apply. In the baseline referring to the DO 178c, it is shown that all the objectives have been applied. For the DO-297 we can see on the corresponding baseline that only apply the objectives mentioned in table A-1 of the DO-297.

7.2.4.3 Integration assurance

The "Avionics_IMA_Reuse" assurance project which focuses on the integration within an IMA architecture. In Avionics_IMA_reuse assurance project references to the objectives mentions on DO-297, table A6 for module reuse. The "AvionicsMACSPlatform" assurance project is modelled as a subproject of the Avionics_IMA_Reuse project. There are two evidence sets for all the artefacts required by the standards explicitly shown on the evidence models. One evidence model is created for evidence associated with DO-178c and another for the DO-297.

In this use case the argumentation is focused on the compliance arguments. Those arguments are automatically generated when creating the assurance project and selecting the parts of the standard the project will comply with. The objectives of the standard are transformed into claims and for this use case these claims have been identified as public, just as the evidences used for compliance.

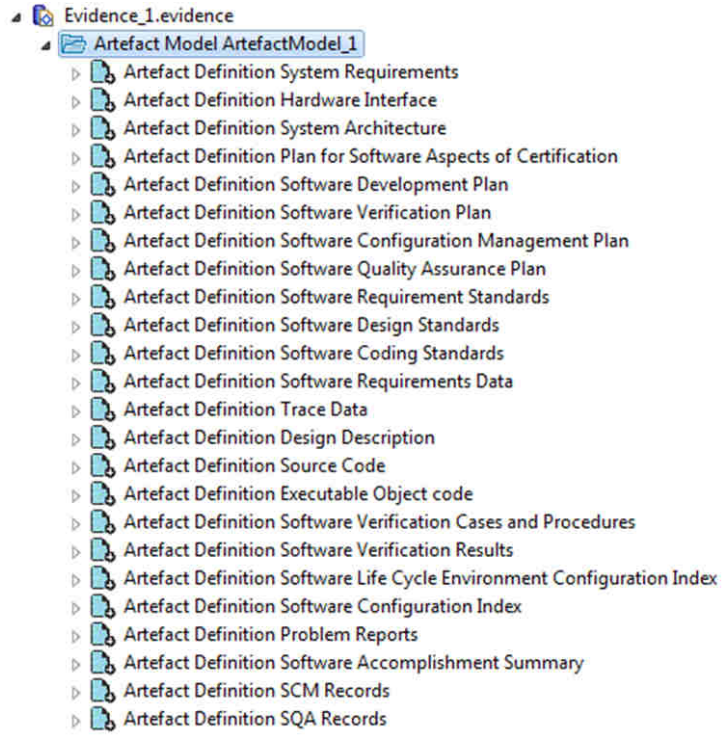


Fig. 90 Evidence model for the Avionics_ExecutionPlatform Project

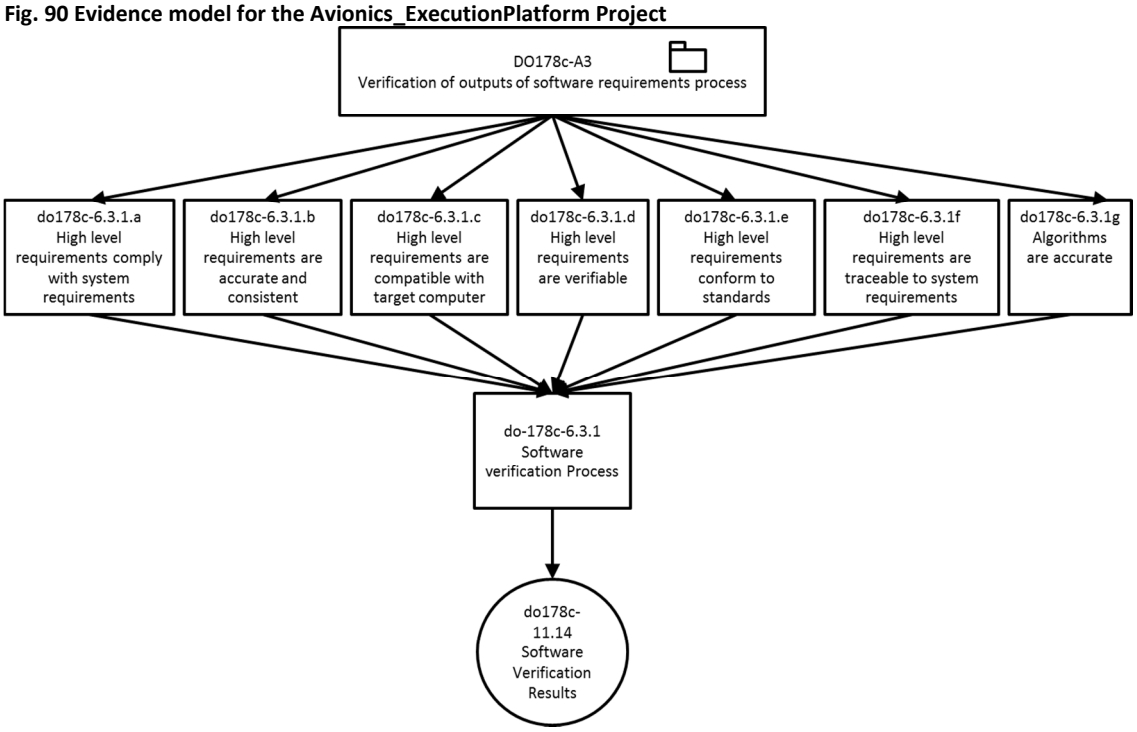


Fig. 91 Excerpt of the compliance argumentation on the Avionics_ExecutionPlatform Project

The Avionics_IMA_Reuse project is focused on the compliance with the DO-297. The baseline includes the activities about the module reuse (Task 6) and from the IMA module development process the objectives:

- Platform integration is complete
- Health monitoring and fault management functions of the IMA platform are provided and documents for use by the hosted applications and the IMA system.
- Quality assurance, configuration management, integration, validation, verification and certification liaison for the module/platform are implemented and completed.

In the evidence model is shown the pieces of evidence created to comply with these objectives.

The argumentation on this assurance project is focused on the feasibility of the reuse. We have taken advantages of the best practices on argumentation by instantiating the reuse argumentation pattern shown in Fig. 62.

7.2.4.4 Assurance Contract

In the contract of this two assurance projects, the objective of the arguments is ensuring that partitioning services will not introduce any interference on the application that will run on the platform.

The information in the usage domain rules provide enough data to translated into the contract. Some of the structured expressions used are the ones presented in Table 17.

Table 17 Structured expressions used in avionics case study

Structured expression	Instantiation on the case study
{fault|failure} is adequately mitigated by {fault accommodation technique}	Communications failure is adequately mitigated by the use of dissimilar redundant communications
{artefact} was produced during {activity}	{Software Requirement Standards} was produced during {Software Planning Process}
{item|system|subsystem|element|component} shows {'partially compliance | full compliance | not compliance} a level of compliance to objective {objective id} for the standard {standard} to level {criticality level}	MACS platform shows { full compliance } a level of compliance to objective {Software Development Processes Objetive 1} for the standard {DO-178C} to level {DAL Al}
{activity} satisfies {requirement}	{Software Requirements Process} satisfies {5.1.2a requirement} of {DO-178C}

7.3 Automotive case study

7.3.1 System Description

The automotive safety assessment process is depicted in Fig. 92. The same structure as in the avionics domain in Fig. 82 is used, describing both the safety assessment as well as the system development process, and the component supplier process is detailed as well.

The key difference between this automotive model and the processes in the other domains are that the automotive domain is characterised by the *absence* of national and international regulators or certification authorities for system safety. Whilst the services of reviewers (such as Independent Safety Assessors (ISAs)) are often used by the vehicle manufacturers (OEMs) and component suppliers, they are always engaged on a commercial rather than quasi-regulatory basis; reviewing practices and exact roles of reviewers vary between different countries and even different vehicle manufacturers' supply chains.

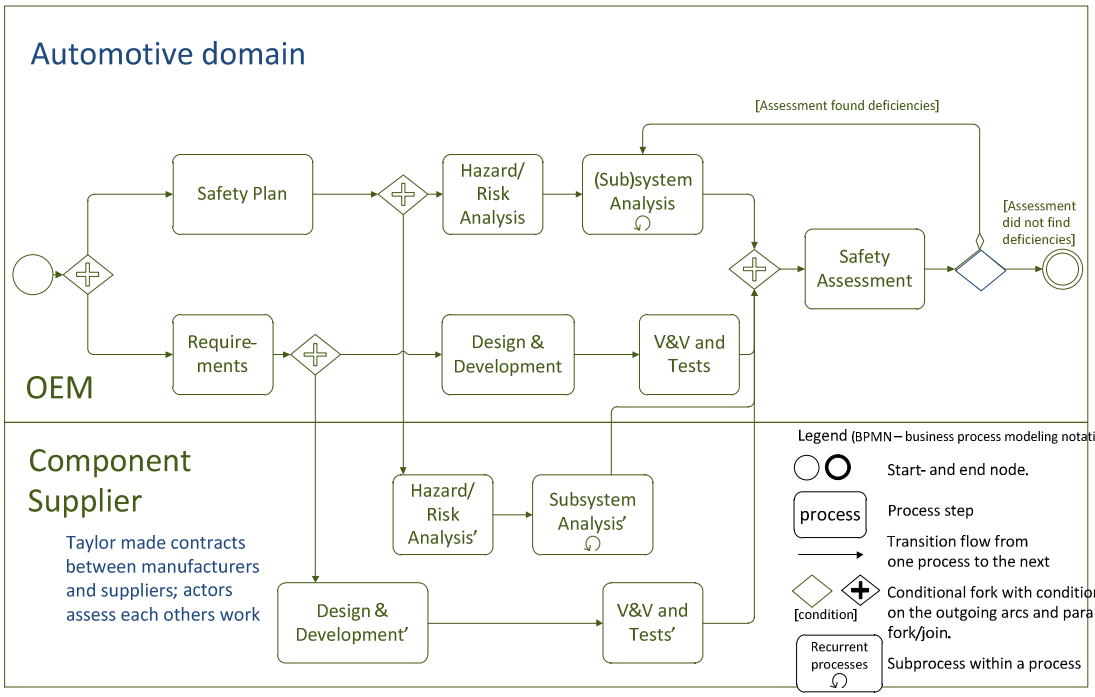


Fig. 92 Safety assessment business process for the automotive domain

7.3.1.1 Industrial case study actors and environment

The system environment is constituted by the electric vehicle interfaces (mechanical, electrical and electronic) and only one actor is involved: the driver.

The man-machine interface of the vehicle dashboard communicates continuously to the driver the status of the gear shift that he has selected last time and, then, also the parking state, if the case. The communication is transmitted by the Vehicle Control Unit (VCU)

(see Fig. 93), which is an electronic board in charge of monitoring the complete status of the vehicle and, in particular for this case, the status of the gear shift.

The parking actuation consists of a mechanical lock on the vehicle gear. When the vehicle is switched off, the only way to alert the driver of the parking actuation (mechanical) is an acoustic alarm (the electrical supply is assured by a direct connection to the 12 V auxiliary battery of the vehicle), whose intervention is triggered when the vehicle door is opened without a parking actuation.

7.3.1.2 Industrial use case operational scenarios

The driver decides to stop the vehicle and, once he has positioned the vehicle in the desired parking place, he must select the Parking mode (normally indicated by a "P" on the gear shift of an electric vehicle) by the corresponding switch on the gear shift. The dashboard signals the actual selection. The driver can open the door and leave the vehicle. If the driver does not select correctly the Parking mode, the opening of the door causes an acoustic alarm that signals to the driver the uncorrected and dangerous condition of the vehicle.

7.3.1.3 Main functions provided by the system

The main function of the electronic parking system is to maintain the transmission of the electric vehicle blocked, avoiding any undesired motion of the wheels when it is stopped for parking.

This function of the system is achieved by the management of the park pawl (mechanical engagement) actuation when the Parking state is entered or exited by the Gear Selector Module logic, respectively when Parking mode has been selected or deselected by the driver.

When the Parking mode is enabled, the torque request sent from the electronic Vehicle Control Unit (VCU) to the power inverter module of the drive train (power inverter + electric motor) is set to zero and the latter is required to remain in torque disabled mode, thus the electric motor cannot receive any electric current able to make it rotate. When this mode is selected, a request is sent to the logic of the electronic parking system to engage the park pawl, thus providing the mechanical locking of the transmission.

7.3.1.4 Architecture of the system

The electronic parking system can be considered as composed of the following elements:

- The electronic parking Control Unit, implementing the high level management logic;
- The "PRND" Switches (Gear Selector Module), implementing the low level software and physically driving the motor which moves the park pawl;
- The Parking Lock System, including mainly the park pawl, the motor for the actuation, the motor position sensor and the park pawl position sensor.

Fig. 93 represents a block diagram of the system and in Fig. 94 a schematic drawing of its mechanical realisation is represented.

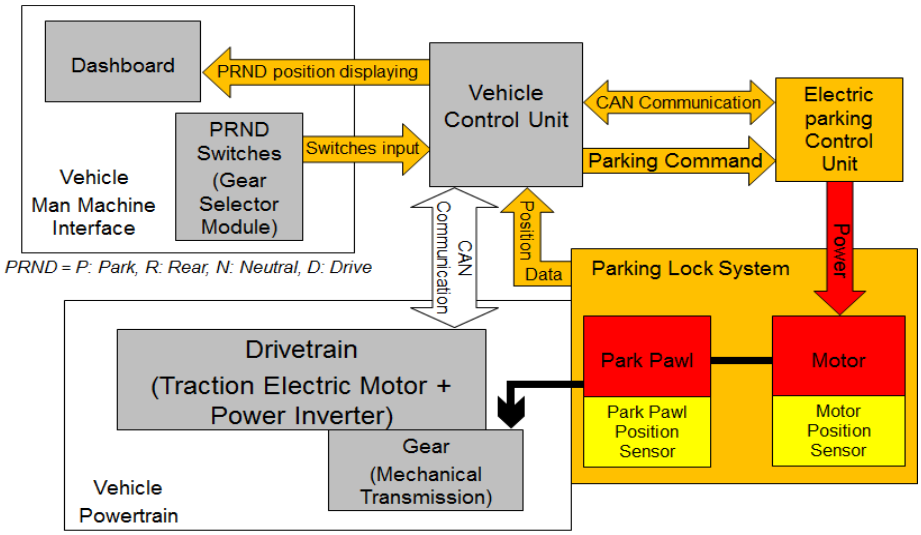


Fig. 93 Electric parking system main blocks

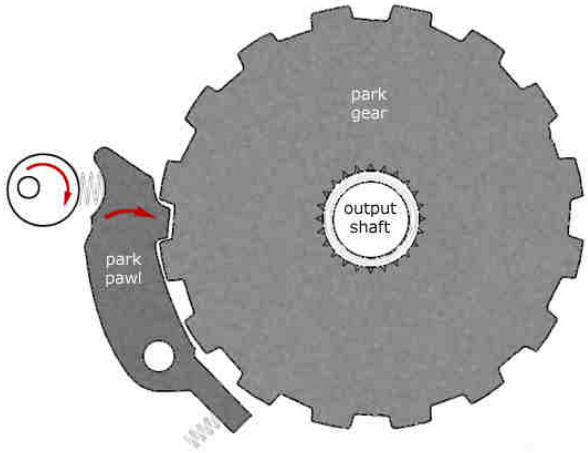


Fig. 94 Schema of the electronic parking lock system

7.3.1.5 General characteristics of the system

There are no particular environmental constraints or performance requirements in terms of actuation, but the park pawl must be mechanically consistent in order to sustain the blockage of the vehicle also in case of a high degree of road slope. The position sensors must be able to guarantee the correct signalling of the effective actuation.

7.3.2 Description of the Compositional Approach

The system considered as SEooC, more specifically, does not reuse component from other contexts, but is itself a component reusable for various contexts.

The electronic parking system is an example of an SEooC application, for which the safety is assured by a closed loop in ISO 26262: a list of assumptions to be used for the application of the system in new contexts is outlined as a part of the item definition and

this list will be verified during the integration of the system in the vehicle. If the assumptions are shown to be invalid, the impact analysis and the related configuration management and change management workflows will support the further modifications of the various safety work products for the envisioned aims.

The SEooC can be viewed as a complementary way with respect to that of the proven in use argument: the second is a new element derived from a known and tested context (vehicle) and has to be integrated in a different context (vehicle), while SEooC is a known system (on the shelf) for an assumed context (vehicle) in which it should be integrated, once the initial assumptions would be shown.

The electronic parking system is a safety element out of context in the sense that it is not developed for a defined electric vehicle, but it is assumed to be compliant to certain characteristics of an electric vehicle for its application (the assumptions). These characteristics are related to the vehicle maximum speed, weight and available sensors signals, and with respect to the electric/electronic basic interface of the dashboard and to the mechanical interface of the transmission. All the vehicles which have the suitable characteristics within a certain limited range (e.g. vehicle speed and weight under a certain maximum value, some standard interfaces for sensor signals and communication data, value of mechanical strength of parking pawl) can integrate this system after the verification of the assumptions.

In Fig. 95 a schematic detail of an example for the SEooC application is reported, derived from the previous figure.

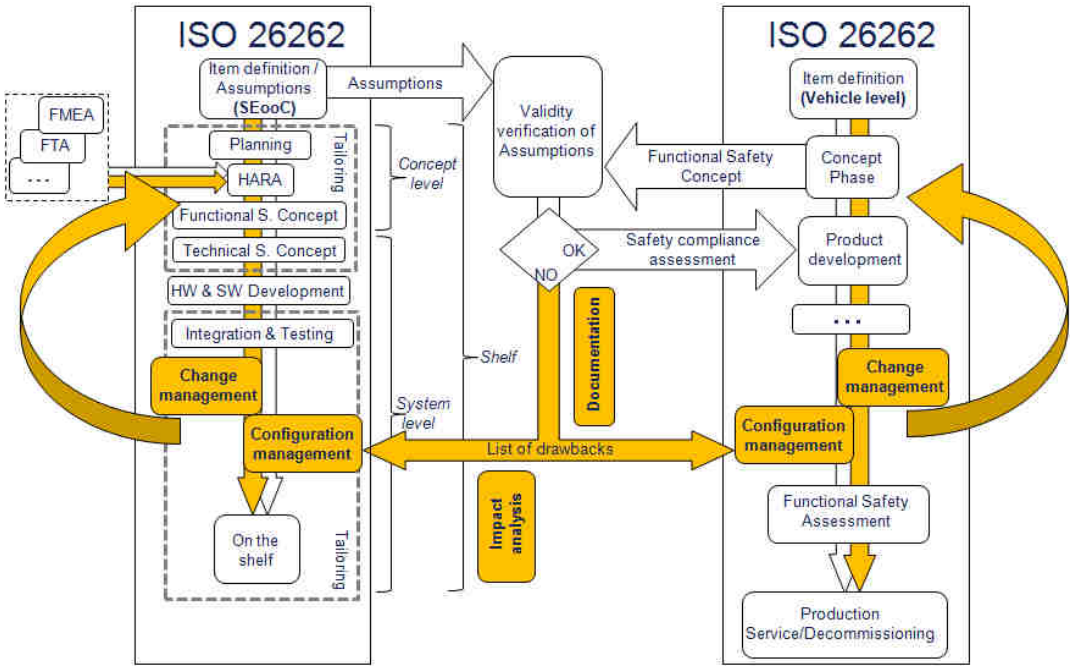


Fig. 95 Example of automotive SEooC use case application

7.3.3 Source data

All the information is generally collected into Word/Excel documents. Life cycle data used from the avionics domain are described in the table hereafter.

Table 18 Automotive Life Cycle Data

Document	Life Cycle Data provided
Item definition	Description of the system, by collecting all the information from the functional point of view with reference to the final user(s)
Hazard Analysis and Risk Assessment	Results from the Hazard Analysis and Risk Assessment, the ASIL definition and the safety goal. It is structured with an Excel table organized mainly in three sheets, which receives in input the information from the "Item definition".
Functional Safety Requirements	It contains the Functional Safety Requirements and the derived technical safety requirements. This constitutes the safety requirements chain that contains the Functional Safety Requirements (FSR) derived from the Safety Goals and the Technical Safety Requirements (TSR) derived from the FSRs. The safety requirements chain is collected into an Excel table, containing also the allocation of the FSRs to the elements of the preliminary architecture and their ASILs derived from the SGs; also the TSRs have their respective ASIL assignment from the corresponding FSRs and their allocation to a more detailed system architecture consequent to TSRs definition.
Vehicle functional safety requirements.xls	Functional safety requirement for the electric park defined at vehicle level
Assumptions.xls	Assumptions about the vehicle made during the SEooC development
ISO 26262	Road vehicles – Functional safety

The matching of FSRs of the vehicle with the FSRs assumed for the electric parking system assures the validity of the assumptions and allows the integration of the SEooC into the target vehicle, for which the technical safety requirements of the electric parking system will be integrated in its technical safety requirements at the product development level.

The above described documentation constitutes the evidences for the project deployment implemented into the platform for the electric parking system use case.

The evidences are supported by argumentations that are part of the content of the reports constituting the work products as required by the standard.

7.3.4 Case study implementation

7.3.4.1 Assurance Compliance Modelling

The functional safety projects development in the automotive domain is conformant to ISO 26262 standard requirements. Our first step has been to create a model of ISO 26262 creating standard to establish the *reference framework*. The objective of this phase is to be able to share a non-ambiguous and formal interpretation of the standard. We have focused on the parts 3 (Safety Concept) and part 4 (Product development at system level) of ISO 26262. We have addressed two top level activities, safety concept phase and product development at system level. Those activities are decomposed into sub activities. Table 19 extracted from the annexes of ISO 26262 will be used as example to show the modelling process.

The elements of column 'Clause' can be mapped as Activity classes. The 'Objectives' column is mapped into the objective parameter of the Activity class. The columns 'Prerequisites' and 'Work products' are easily mapped as artefacts in our meta-model.

Activities might need to fulfil requirements on how they should be done. For example the clause 7.4 Requirements and recommendations from ISO 26262 includes elements that are mapped as requirements that should be fulfilled by the activity: Hazard analysis and risk assessment.

Table 19 Excerpt of table A.1 from ISO 26262 annexes [ISO 26262].

Clause	Objectives	Prerequisites	Work products
5. Item definition	The first objective is to define and describe the item, its dependencies on and interaction with the environment and other items. The second objective is to support an adequate understanding of the item so that the activities in subsequent phases can be performed	None	5.5 Item definition

Clause	Objectives	Prerequisites	Work products
6 Initiation of the safety lifecycle	The first objective of the initiation of the safety lifecycle is to make the distinction between a new item development and a modification to a existing item (see ISO 26262-2:2011, Figure 2) The second objective is to define the safety lifecycle activities (see ISO 26262-2:2011, Figure 2) that will be carried out in the case of a modification.	Item definition	6.5.1 Impact analysis 6.5.2 Safety plan (refined)
7 Hazard analysis and risk assessment	The objective of the hazard analysis and risk assessment is to identify and to categorise the hazards that malfunctions in the item can trigger and to formulate the safety goals related to the prevention or mitigation of the hazardous events, in order to avoid unreasonable risk.	Item definition	7.5.1 Hazard analysis and risk assessment 7.5.2 Safety goals 7.5.3 Verification review report of the hazard analysis and risk assessment and the safety goals
8 Functional safety concept	The objective of the functional safety concept is to derive the functional safety requirements, from the safety goals, and to allocate them to the preliminary architectural elements of the item, or to external measures. Item definition	Hazard analysis and risk assessment Safety goals	8.5.1 Functional safety concept 8.5.2 Verification report of the functional safety concept

Artefacts in our model are mapped with work products from the standard. So the activity Hazard analysis and risk assessment will produced the HARA report such at is described on the standard ISO 26262 on clause 3-7.5.1. The work products sometimes are required to include some sections or information. In this case, we model them as requirements that constrain a specific artefact. (See Fig. 96)

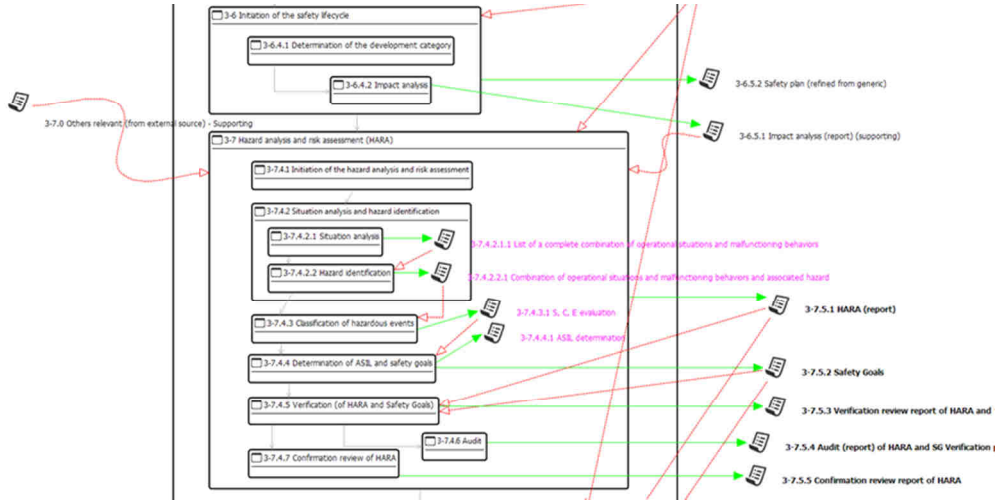


Fig. 96 Excerpt of the ISO 26262 model

7.3.4.2 Assurance Modeling for Platform/Component

Once we have modelled the standard, the component safety manager will follow the process showed in Fig. 97. We create the assurance project for our SEooC, the parking system. In the assurance project the structure of the project, a baseline and the conformance related argumentation are generated.

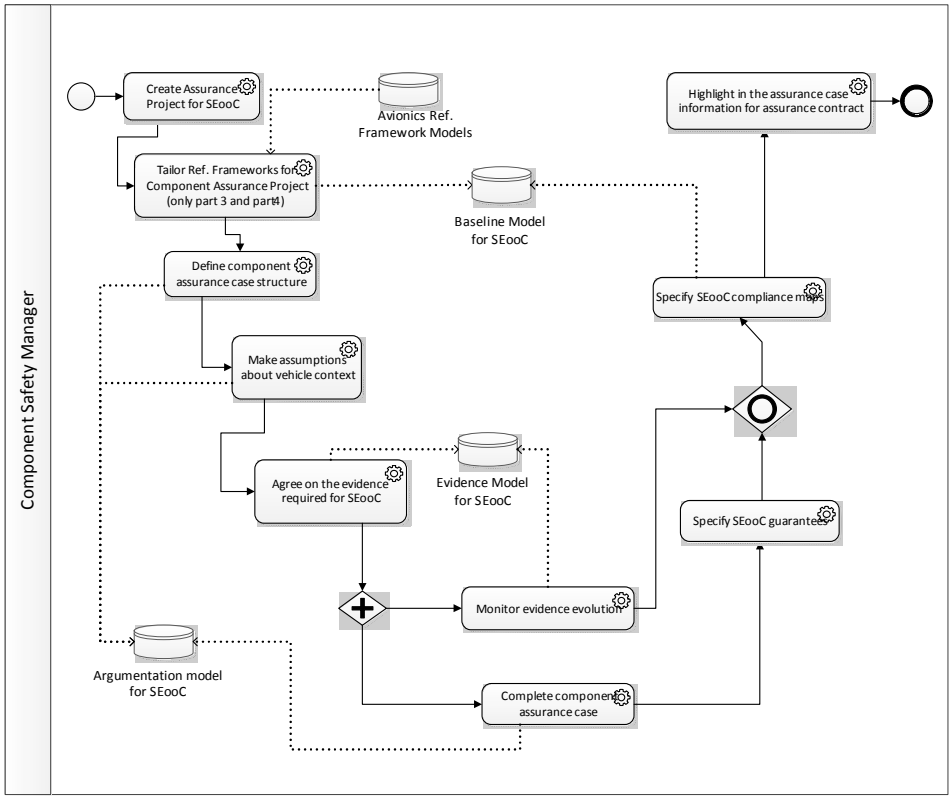


Fig. 97 Process followed by the SEooC responsible

For the SEooC assurance project we have tailored the baseline, the part of the standard that applies to this case. In Fig. 98 we summarize our interpretation of being compliant with the SEooC related activities.

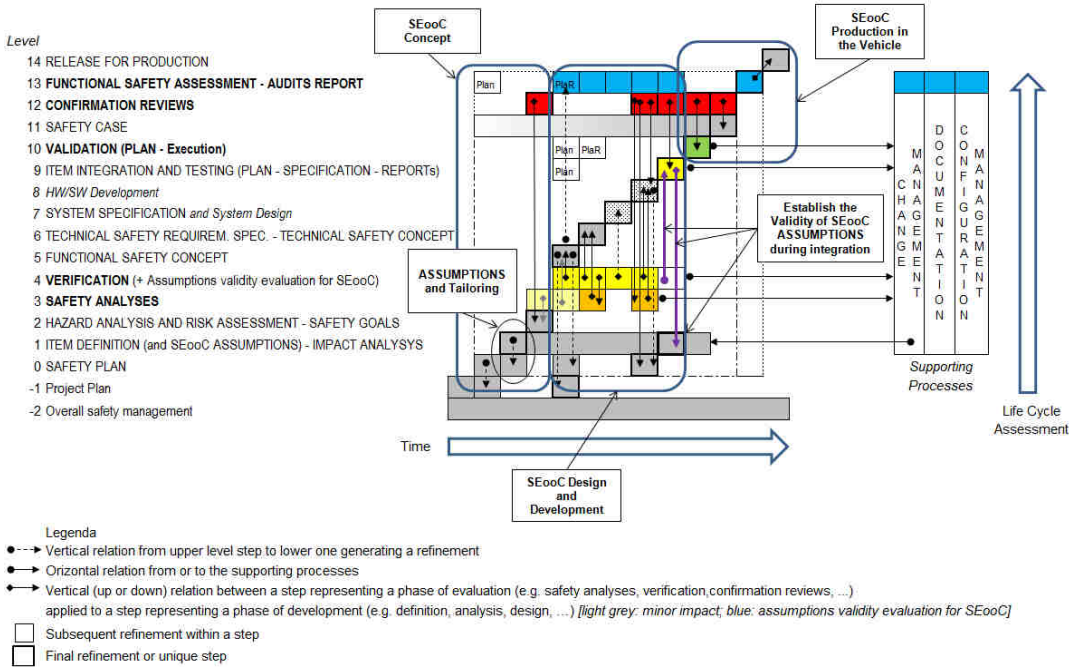


Fig. 98 Interpretation of the ISO 26262 for the SEooC development

The 'level' label is a degree of evolution in the lifecycle assessment towards the whole set of required evidence, where its value represents a baseline for implementation. Each degree of evolution is built based on its previous baseline; but in some cases the requirement in some higher baseline can cause a revision (refinement) of a requirement in a lower baseline (e.g. item definition versus safety plan). Some requirements that are recursive or are simple 'refinements' belong to the same level, even if they are performed subsequently (e.g. refinements of the safety plan belong to the same level: the safety plan level).

In this project not all the complete ISO 26262 applies but just some part; we have focused on part 3 and 4 of the standard.

In the SEooC assurance project we have two types of argumentation created. One part is the automatic generated argumentation. This generation is created by a model transformation. All the activities and requirements selected on the baseline are transformed into claims, the squares and the reference artefacts resulting for an activity are converted into information elements, the circles. The other part indicates:

- The public claims about the functionality offered by the SEooC.
- Assumptions made about the vehicle

We also have an evidence model with all the evidences produced for the SEooC compliancy. The electric parking system evidence model is solved by the evidences represented by the content of the documents described in the previous section (7.3.3): “Item definition” (containing the SEooC assumptions), “HARA” report, List of Safety Goals and of the Functional Safety Requirements, together with the list of vehicle Functional Safety Requirements. All these elements are traces on the evidence model as it is shown in Fig. 99.

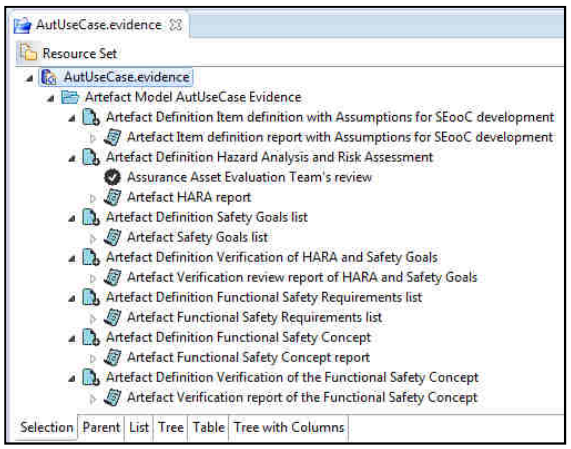


Fig. 99 Assurance project evidence section

Finally, at the SEooC baseline we have created the traces for the compliance using the mapping setting feature. At the baseline editor, we have available the compliance mapping menu as it is showed in the following figure, on which the available evidences (as “artefact models”) can be linked to the nodes of the baseline previously generated.

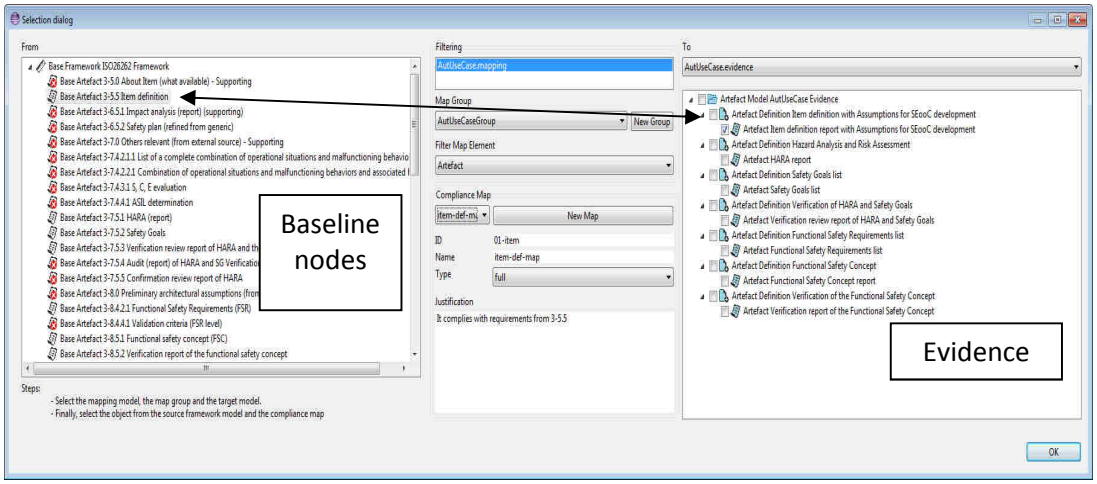


Fig. 100 Mapping window.

7.3.4.3 Integration Assurance

For the composition approach we have created a vehicle assurance project that deals with SEooC integration. A sub-project represents the SEooC development which we have included information about how it has been developed. A “mother” project represents

the vehicle assurance project. The vehicle level project is a black box; the information about how the vehicle is developed is not visible for us, it is only available information about the integration.

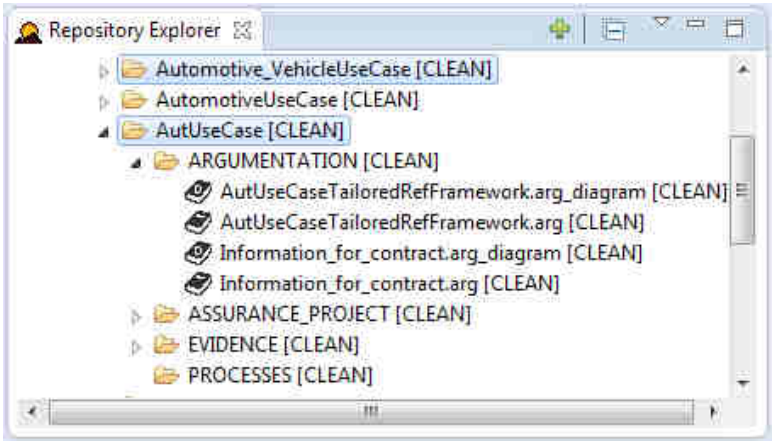


Fig. 101 View of the assurance projects created for the case study

In the vehicle assurance project we need to reference to elements that have been created on the SEooC assurance project. This way on the integration baseline we reference to activities such as Specification of assumption on SEooC functional safety requirements and establishment of validity of them with respect to the target vehicle functional safety requirements.

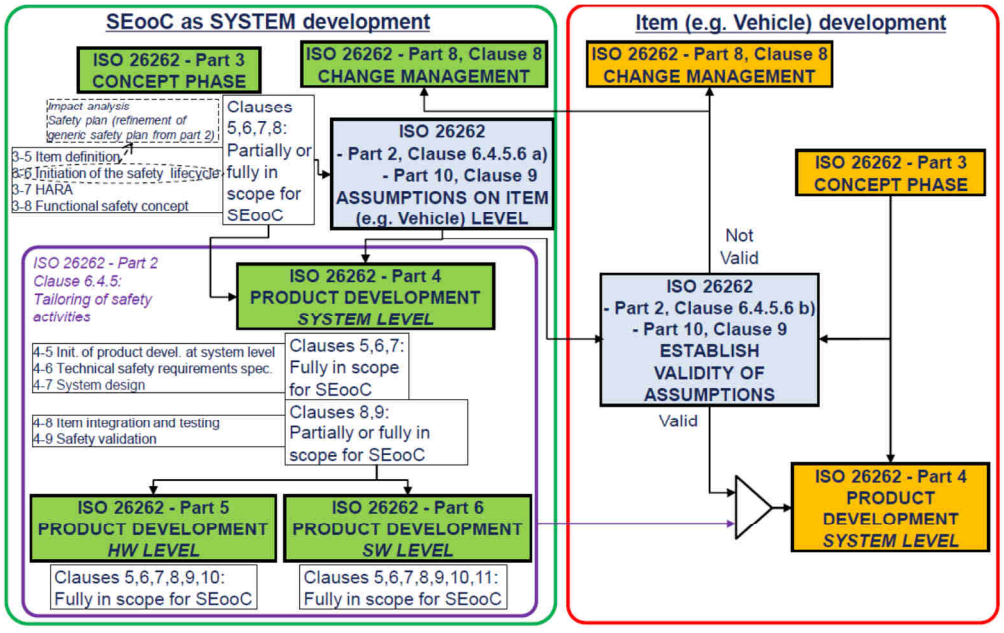


Fig. 102 Process for the SEooC integration within the vehicle

As the vehicle level information is considered a black box, on the baseline we only see those activities related to the integration of the SEooC.

7.3.4.4 Assurance contract

In the argumentation we only see the context of the vehicle and the functional requirements expected for the SEooC. We also have created a contract that links the assumptions and public claims or guarantee made on the SEooC assurance project that are represented on the argumentation, with the functional requirements and context from the vehicle which are also shown on the vehicle argumentation.

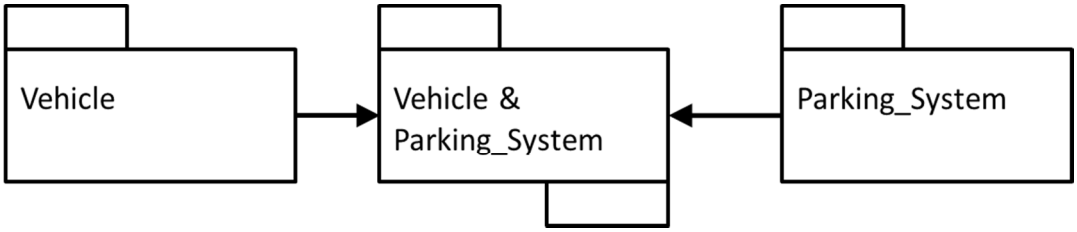


Fig. 103 Contract representation between the vehicle and the electric parking system

Some of the structured expressions used for the assumptions and guarantees definition can be seen on the following table.

Table 20 Structured Expressions used for the assumptions and guarantees of the automotive assurance contract

Structured expression	Instantiation on the case study
{fault(s)} are diagnosed by {syselement} and indicated by {action}	{VCU internal failure(s) [SW error(s), Electrical fault(s)]} are diagnosed by {parking system} and indicated by {setting an error status flag and transmit it on CAN network}.
{syselement} diagnoses {conditions}	{VCU} diagnoses {the failures on Electric Parking System button switches and related wired connections}.
{failure} is indicated by {symptom:condition} and detected by {syselement}	{GSM shut down} is indicated by {loss of communication} and detected by {VCU}
{conditionexpression} are indicated by {action}	{faults diagnosed by GSM} are indicated by error state flag transmitted to VCU via CA
{syselement} will perform {action} when {condition}	{parking system} will perform {Engagement command cancellation} when {engagement command is affected by an error}
{syselement} will perform {action} to maintain {state} when {condition}	{the VCU} will perform {the handshake process invalidation and the GSM let park pawl to stay in its current position (if current position is park pawl engaged or unknown disable traction)} when {a parking pawl motor error is detected during VCU to GSM handshake}

{action} will be performed to (maintain | achieve | prevent | avoid) {state} when {condition}	{A disengagement command cancellation} will be performed to {maintain {the parking pawl engaged} when {al error affect the disengagement}.
{state} will be (maintained | achieved | prevented | avoided) when {condition}	{The parking button LED set on/off} will be {achieved} when {the parking pawl is engaged/ disengaged}.
{systemelement} has { attribute|{property} that does {action} | simpleproperty}	{Vehicle electric parking system} has {the network signalling by CAN bus}.
{syselement} allows {action}	{Cockpit and charge port lid of the vehicle} allows {a stable visualization of a label}.
{agent} will {action} {condition}	{Driver} will {block the vehicle} { before leaving it in charging} + supporting claim: Label in charge port lid reminds driver to block vehicle"

7.4 Medical devices case study

7.4.1 System Description

The case study focuses on the development of the medical device *Automated External Defibrillator* (AED), a current research trend inside NUTES. NUTES is part of an initiative for promoting the technological development of Brazil, where the Brazilian Health Ministry has started some technological transfer projects from well-consolidated manufacturers to institutes for science and technologies in order to retain the know-how of manufacturing medical devices inside the country. In this context, the NUTES project is in charge of receiving and improving methodologies for manufacturing AEDs from the Lifemed and providing new improvements.

AEDs are consolidated as a therapy for the ventricular fibrillation/tachycardia, which are the cardiac arrhythmias with highest incidences of fatal cases. In the treatment of such conditions, any delay in the application of the defibrillator shock is an important issue for investigation, since each minute without the shock implies in a loss among 7% to 10% of the chance of surviving. The usage of AEDs has gained much more popularity, since they can be used even without a specialized rescuer team available.

7.4.1.1 Industrial use case actors and environment

Fig. 106 presents the essential parts of the AED system as a context diagram. Considering the safety case, we achieve the goal of showing each external entity, the main functional unities and their interaction with the system. The main input variable to be received is the cardiac pulses of the Patient. A module defined as Signal Analyzer uses of sophisticated algorithms for detecting the signal complexity and decide if a defibrillator pulse is necessary in case of fibrillation. If it is the case, the Shock Generator is in the

responsibility of controlling the main output variable, the energy by providing it in the Biphasic Truncated Exponential waveform to the Patient chest through the Pads.

7.4.1.2 Industrial use case operational scenarios

The patient suffers a tachycardia losing the heart beat signal. The rescuer put the AED's pads on the patient chest and delivers the shock. The AED system must detect ventricular fibrillations through the ECG signal of the patient. Fig. 104 presents the essential parts of the AED system as a context diagram. Considering the safety case, we achieve the goal of showing each external entity, the main functional unities and their interaction with the system. The main input variable to be received is the cardiac pulses of the Patient. A module defined as Signal Analyzer employees sophisticated algorithms for detecting the signal complexity and decides if a defibrillator pulse is necessary in case of fibrillation. If it is the case, the Shock Generator is in the responsibility of controlling the main output variable, the energy by providing it in the Biphasic Truncated Exponential waveform to the Patient chest through the Pads. The time for generating the energy should be as low as possible.

In Fig. 104 on the left presents the normal operating AED use case. On the right, the figure describes the main failures that could prevent the normal operation of the system.

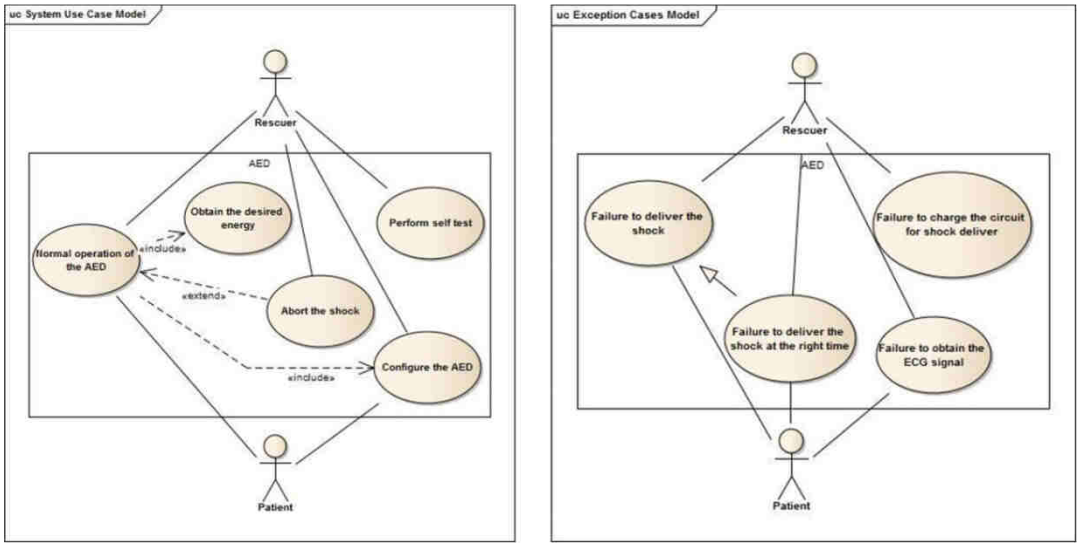


Fig. 104 AED use cases models

7.4.1.3 Main functions provided by the system

The system is implemented using model-based design. All the descriptions and figures are extracted from the model of the system.

The functional requirements from the system are described in Fig. 105.

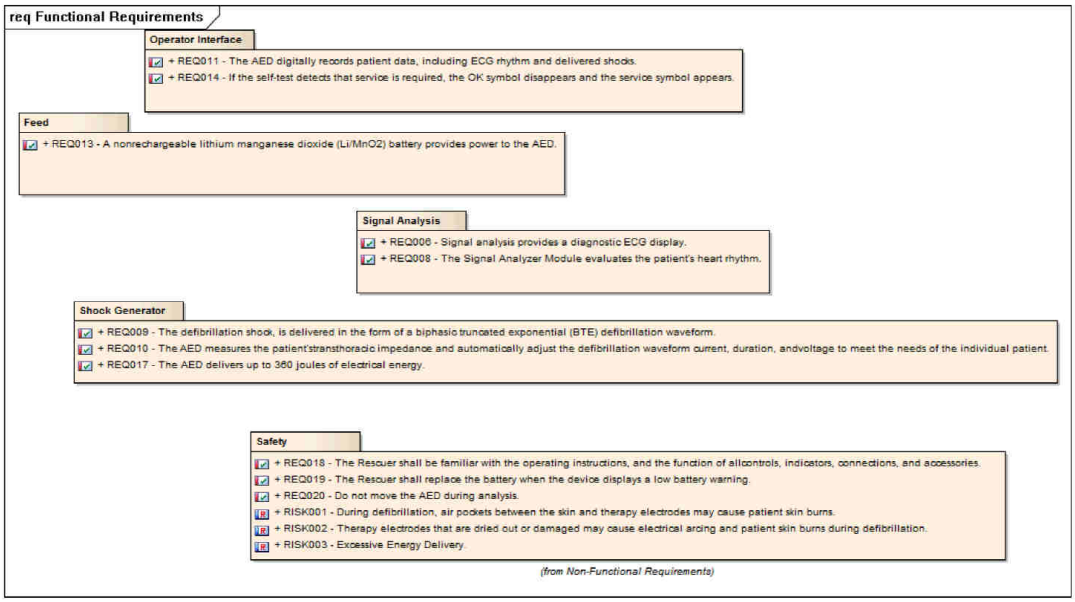


Fig. 105 AED functional requirements

7.4.1.4 Architecture of the system

The AED interacts with two entities that are outside of the system scope:

- The Rescuer that is the operator of the AED;
- The Patient that is the person who had the sudden cardiac arrest.

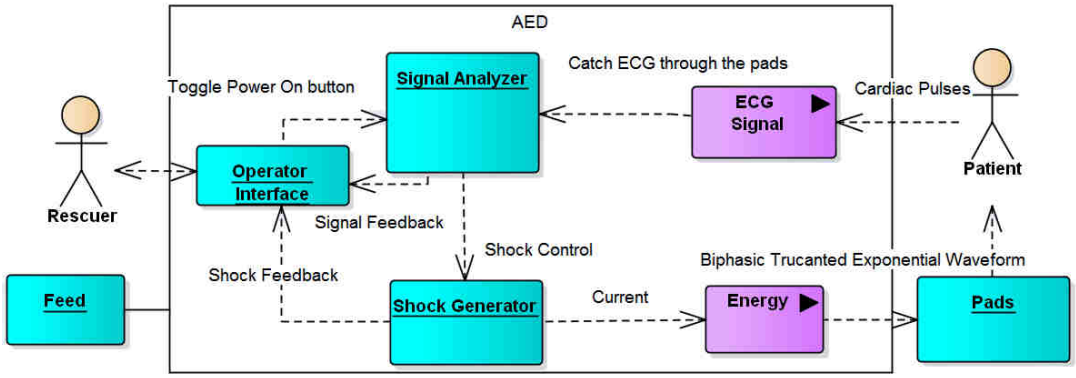


Fig. 106 AED context use case with the main elements that are part of the system

The AED has three entities inside it which are in responsibility of providing it features:

1. The operator interface which is responsible for the Power On button and Shock button, LCD to display information, alarms, sounds, etc.
2. The signal analyzer which has the algorithms to extract the cardiac frequency, signal complexity and decides if there is fibrillation or not.

3. The shock generator, which has the charge circuit in order to obtain the necessary energy, guaranteeing the isolation of high voltage circuits. It also manages the discharge of this energy to a load.

In Fig. 107 the main architecture blocks in which the AED is decomposed are presented.

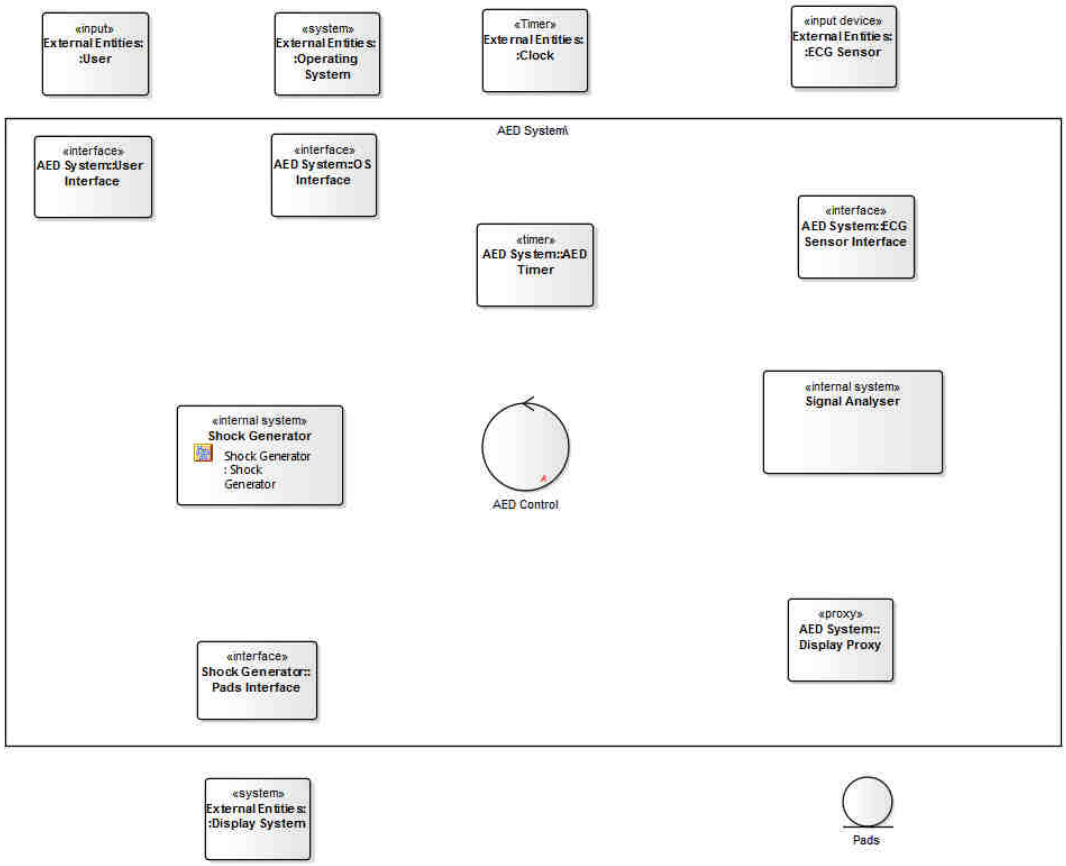


Fig. 107 AED architecture internal blocks

7.4.1.5 General characteristics of the system

AED provides a controlled deliver of energy for the patient chest. AEDs are used extensively as a rescue tool for people arrested in cardiac attacks during their daily routine. The purpose of AED is to analyse the ECG signal of the patient and decide whether the shock is necessary or not. Therefore, the tool provides a very quick charging process allowing the discharge to the patient chest in the Biphasic Truncated Exponential waveform. It is operated by a rescuer that just needs to connect the pads, toggle the power on button and perform CPR for 5 cycles of 30 compressions to 2 breaths after the shock application.

7.4.2 Description of the Compositional Approach

The AED is composed by two main components the Signal Analyser component and the Shock Generator component. The case study is focused on the integration of the Signal

Analyser software which has been developed as a critical component into the AED system.

From the AED system the Signal Analyser component is consider a SOUP. Term used in the IEC 62304 standard to refer to software components of unknown precedence.

In Fig. 107 the overview of software development processes for the software development is explained. The composition approach will focus on the software integration and verification with the final AED target.

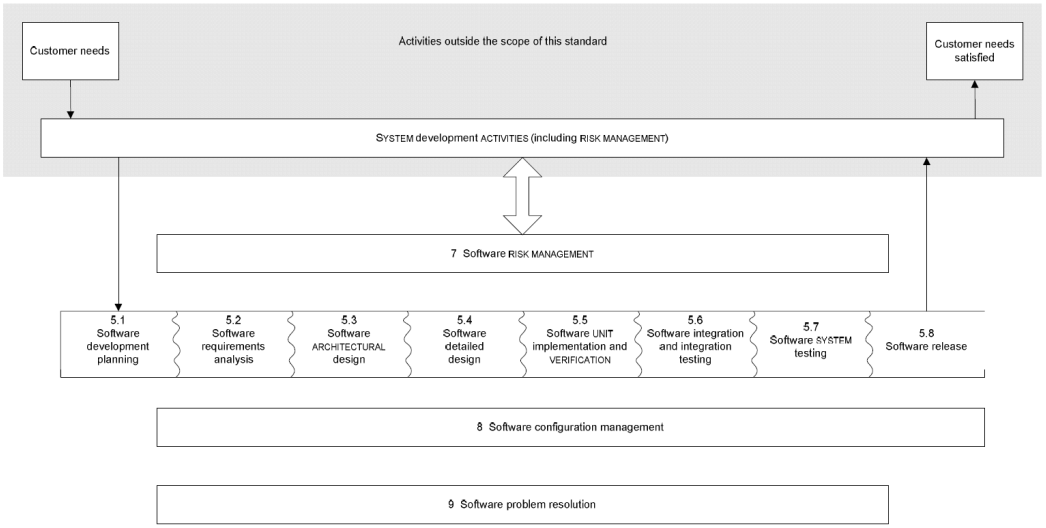


Fig. 108 IEC 62304 Overview of software development PROCESSES and ACTIVITIES

In Fig. 109 the main safety requirements are decomposed. Apart from that a high level safety case has been previously developed using GSN only about the shock generator component. During the case study this will be enhanced with arguments about the safe composition.

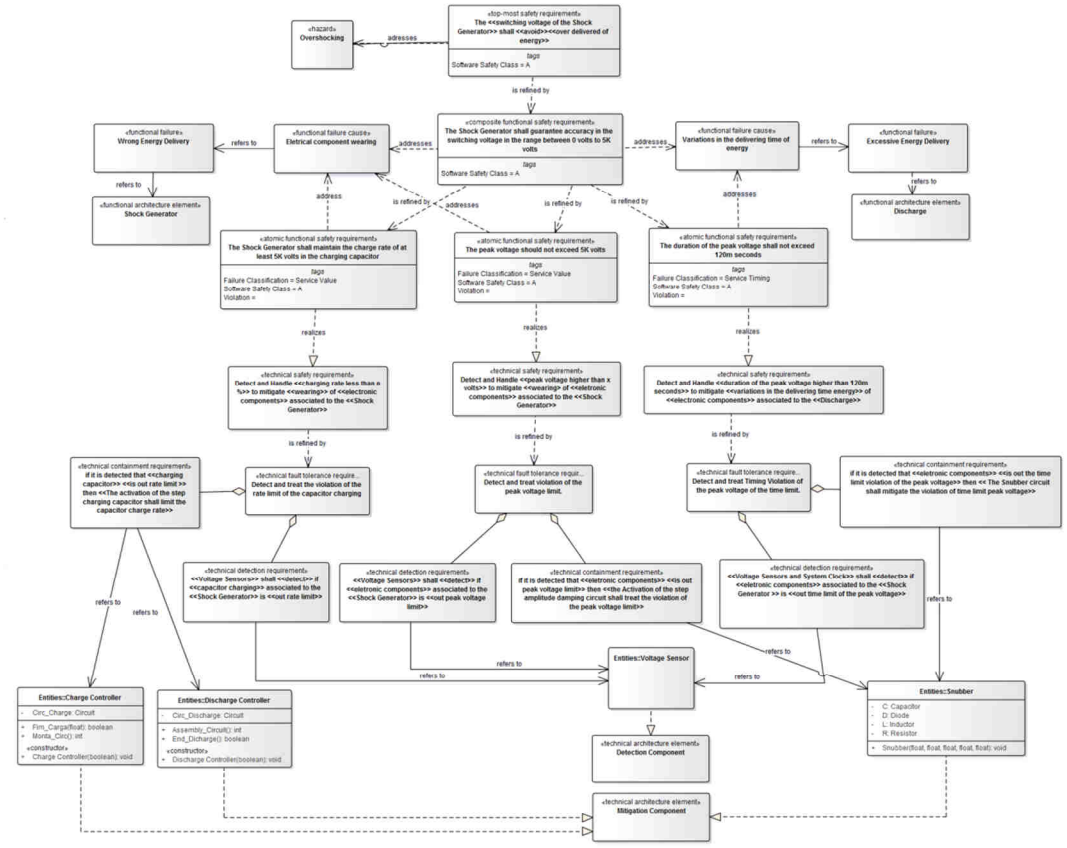


Fig. 109 Safety requirements decomposition [Antonino et al. 2015].

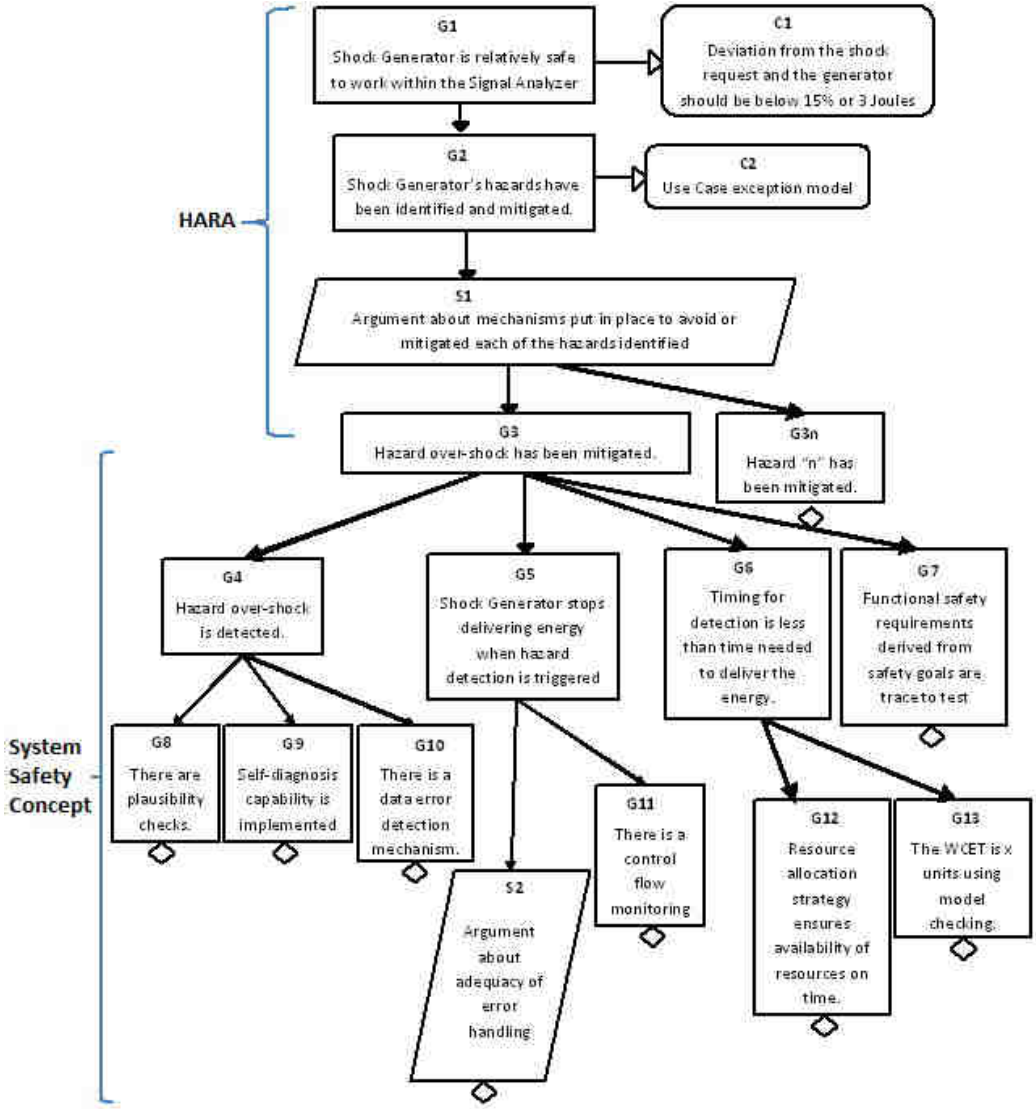


Fig. 110 Excerpt of the shock generator component safety case

7.4.4 Source Data

The development of the system has been done in parallel with the assurance work. The system has been developed using model-based design and documents to comply with the standard have been explicitly created in order to follow the compositional assurance methodology proposed. Engineering data has been available to the assurance team at the same time as the design team and communication has been fluent between the teams. No data from previous certification experiences to comply with the standards was available.

The model used has been AED.eap model using Enterprise Architecture.

7.4.5 Case study implementation

7.4.5.1 Assurance Compliance Modelling

Information Source used are IEC 62304 and ISO 14971. We have modelled both standards into two different reference frameworks. One of the main problems for doing so is that the documentation that should be presented for certification is not explicitly defined. The steps to follow in order to execute a process have been modelled as sun activities. Refinements of an artefact have been modelled as new artefacts.

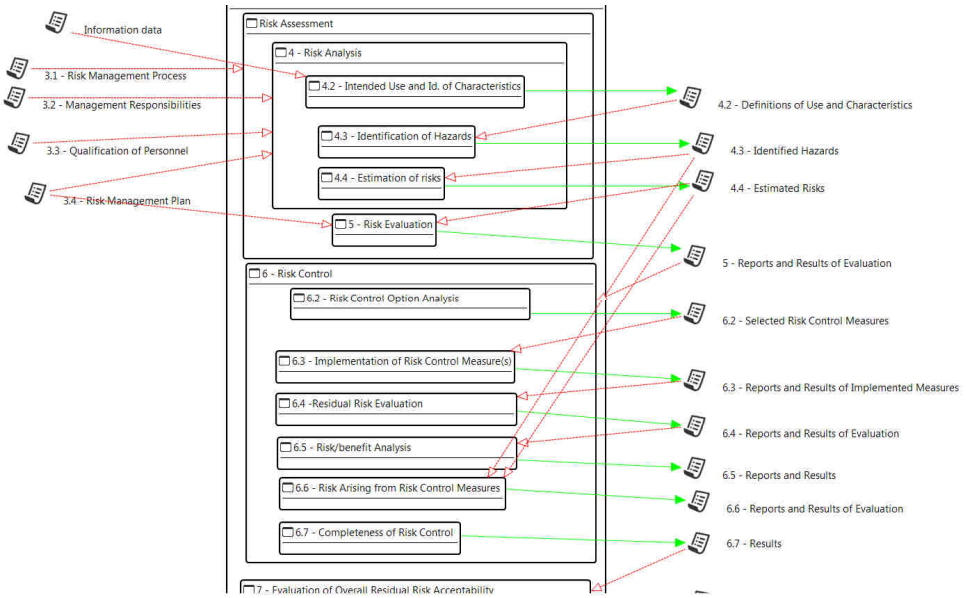


Fig. 111 Excerpt of the ISO 14971 modelization

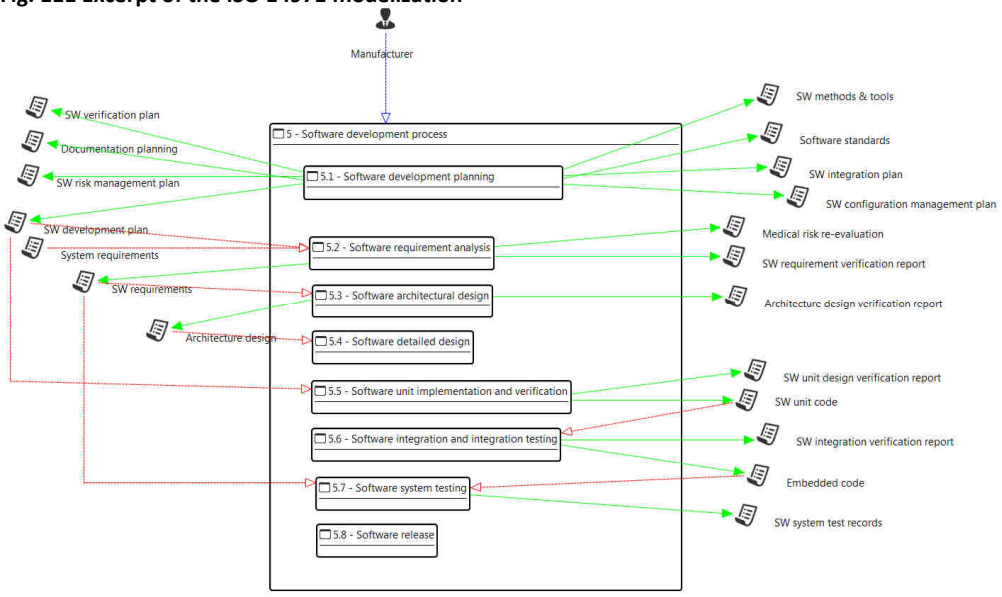


Fig. 112 IEC 62304 graphically modelled

- iec62304.refframework
  - Ref Framework IEC 62304 standard
    - Ref Activity 5 - Software development process
      - Ref Activity 5.1 - Software development planning
        - Ref Requirement 5.1.1 Software development plan
        - Ref Requirement 5.1.2 keep software development plan updated
        - Ref Requirement 5.1.3 Software development plan reference to system design and development
        - Ref Requirement 5.1.4 Software development standards, methods and tools planning
        - Ref Requirement 5.1.5 Software integration and integration testing planning
        - Ref Requirement 5.1.6 Software VERIFICATION planning
        - Ref Requirement 5.1.7 Software RISK MANAGEMENT planning
        - Ref Requirement 5.1.8 Documentation planning
        - Ref Requirement 5.1.9 Software configuration management planning
        - Ref Requirement 5.1.10 Supporting items to be controlled
        - Ref Requirement 5.1.11 Software CONFIGURATION ITEM control before VERIFICATION
      - Ref Activity 5.2 - Software requirement analysis
        - Ref Requirement 5.2.1 Define and document software requirements from SYSTEM requirements
        - Ref Requirement 5.2.2 Software requirements content
        - Ref Requirement 5.2.3 Include RISK CONTROL measures in software requirements
        - Ref Requirement 5.2.4 Re-EVALUATE MEDICAL DEVICE RISK ANALYSIS
        - Ref Requirement 5.2.5 Update SYSTEM requirements
        - Ref Requirement 5.2.6 Verify software requirements
      - Ref Activity 5.3 - Software architectural design
        - Ref Requirement 5.3.1 Transform software requirements into an ARCHITECTURE
        - Ref Requirement 5.3.2 Develop an ARCHITECTURE for the interfaces of SOFTWARE ITEMS
        - Ref Requirement 5.3.3 Specify functional and performance requirements of SOUP item
        - Ref Requirement 5.3.4 Specify SYSTEM hardware and software required by SOUP item
        - Ref Requirement 5.3.5 Identify segregation necessary for RISK CONTROL
        - Ref Requirement 5.3.6 Verify software ARCHITECTURE
      - Ref Activity 5.4 - Software detailed design
        - Ref Requirement 5.4.1 Refine SOFTWARE ARCHITECTURE into SOFTWARE UNITS
        - Ref Requirement 5.4.2 Develop detailed design for each SOFTWARE UNIT
        - Ref Requirement 5.4.3 Develop detailed design for interfaces
        - Ref Requirement 5.4.4 Verify detailed design
      - Ref Activity 5.5 - Software unit implementation and verification
        - Ref Requirement 5.5.1 Implement each SOFTWARE UNIT
        - Ref Requirement 5.5.2 Establish SOFTWARE UNIT VERIFICATION PROCESS
        - Ref Requirement 5.5.3 SOFTWARE UNIT acceptance criteria
        - Ref Requirement 5.5.4 Additional SOFTWARE UNIT acceptance criteria
        - Ref Requirement 5.5.5 SOFTWARE UNIT VERIFICATION
      - Ref Activity 5.6 - Software integration and integration testing
        - Ref Requirement 5.6.1 Integrate SOFTWARE UNITS
        - Ref Requirement 5.6.2 Verify software integration
        - Ref Requirement 5.6.3 Test integrated software
        - Ref Requirement 5.6.4 Integration testing content
        - Ref Requirement 5.6.5 Verify integration test procedures
        - Ref Requirement 5.6.6 Conduct regression tests
        - Ref Requirement 5.6.7 Integration test record contents
        - Ref Requirement 5.6.8 Use software problem resolution PROCESS
      - Ref Activity 5.7 - Software system testing
        - Ref Requirement 5.7.1 Establish tests for software requirements
        - Ref Requirement 5.7.2 Use software problem resolution PROCESS
        - Ref Requirement 5.7.3 Retest after changes
        - Ref Requirement 5.7.4 Verify SOFTWARE SYSTEM testing
        - Ref Requirement 5.7.5 SOFTWARE SYSTEM test record contents
      - Ref Activity 5.8 - Software release
        - Ref Requirement 5.8.1 Ensure software VERIFICATION is complete
        - Ref Requirement 5.8.2 Document known residual ANOMALIES
        - Ref Requirement 5.8.3 EVALUATE known residual ANOMALIES
        - Ref Requirement 5.8.4 Document released VERSIONS
        - Ref Requirement 5.8.5 Document how released software was created
        - Ref Requirement 5.8.6 Ensure activities and tasks are complete
        - Ref Requirement 5.8.7 Archive software
        - Ref Requirement 5.8.8 Assure repeatability of software release
      - Ref Artefact SW verification plan
      - Ref Artefact Documentation planning
      - Ref Artefact SW risk management plan
      - Ref Artefact SW development plan
      - Ref Artefact System requirements
      - Ref Artefact SW requirements
      - Ref Artefact Architecture design
      - Ref Artefact SW methods & tools
      - Ref Artefact Software standards
      - Ref Artefact SW integration plan
      - Ref Artefact SW configuration management plan
      - Ref Artefact Medical risk re-evaluation
      - Ref Artefact SW requirement verification report
      - Ref Artefact Architecture design verification report
      - Ref Artefact SW unit code
      - Ref Artefact SW unit design verification report

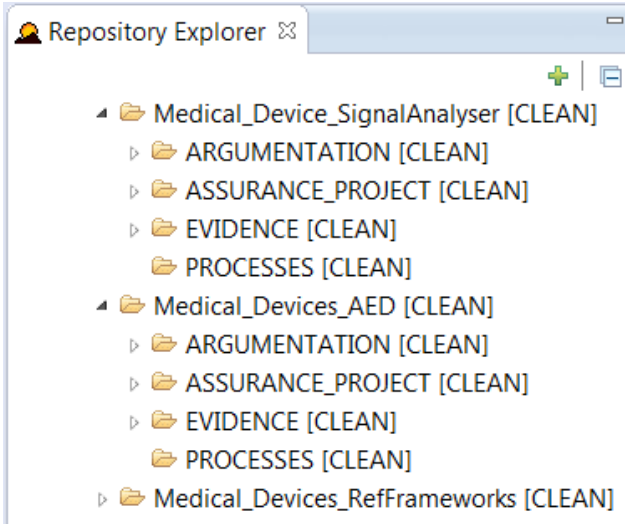
Fig. 113 IEC 62304 modelization on a tree view.

7.4.5.2 Assurance Modelling for Platform/Component

Once we have modelled the standards, we create the assurance project for our Signal Analyser SOUP. This assurance project has two baselines one referring to the IEC 62304 conformance and another one for the ISO 14970 conformance. Concerning IEC 62304 the activities: 5.6 - Software integration and integration testing and 5.7 - Software system testing where not executed and have been left for the AED System assurance project. For the Signal Analyser component two set of evidences have been modelled, one in relation with the ISO 14971 and another one in relation with the IEC 62304. Two argument diagrams have been created for the component, one the auto-generated conformance argumentation and another one specifically focus on the safety requirements allocation and safety mechanism put in place on the component.

7.4.5.3 Integration Assurance

The AED system assurance project includes the integration activities and integrates the artefacts from the Signal Analyser components into its own artefacts. The "Medical_Devices_AED" assurance project includes the "Medical_Device_SignalAnalyser as a subproject.



Both assurance projects are white boxes and the evidences from the Signal Analyser are part of the AED evidences.

The AED argumentation is focused on product-based argumentation and compliance with the standards.

7.4.5.4 Assurance Contract

The assurance contract links the expected functionality and safety mechanism to be implemented by the other module.

Some of the structured expressions used for the assumptions and guarantees definition can be seen in Table 21.

Table 21 Structured Expressions used for the assumptions and guarantees of the AED assurance contract

Structured expression	Instantiation on the case study
{item|system|subsystem|element|component} performs {function}	{signal analyser module} performs {patient's heart rhythm evaluation}
{item|system|subsystem|element|component} will perform {function} when {condition|stimulus}	{signal analyser module} will perform {diagnostic ECG visualization} when {ECG analysis has been executed}
{fault|failure} is diagnosed by {item|system|element|component|agent} and indicated by {action}	{RAM section memory failure} is diagnosed by {signal analyser module} and indicated by {low level drivers test}
{item|system|subsystem|element|component} will perform {action} to avoid {state} when {condition|stimulus}	{signal analyser module} will perform {a CPU test} to avoid {a failure state} when {the period t expires}
{fault|failure} is indicated by {condition} and detected by {item|system|element|component|agent}	{CPU and RAM test failures} are indicated by {system failure alarm} and detected by {signal analyser module}
{agent} will {action}{condition}	{Stakeholders} will {ensure that the AED complies with intended local characteristics}

In some cases the connection between the guarantees and the assumptions have not been one to one and need the rational part of the contract to ensure the validity as the excerpt shown in Fig. 114.

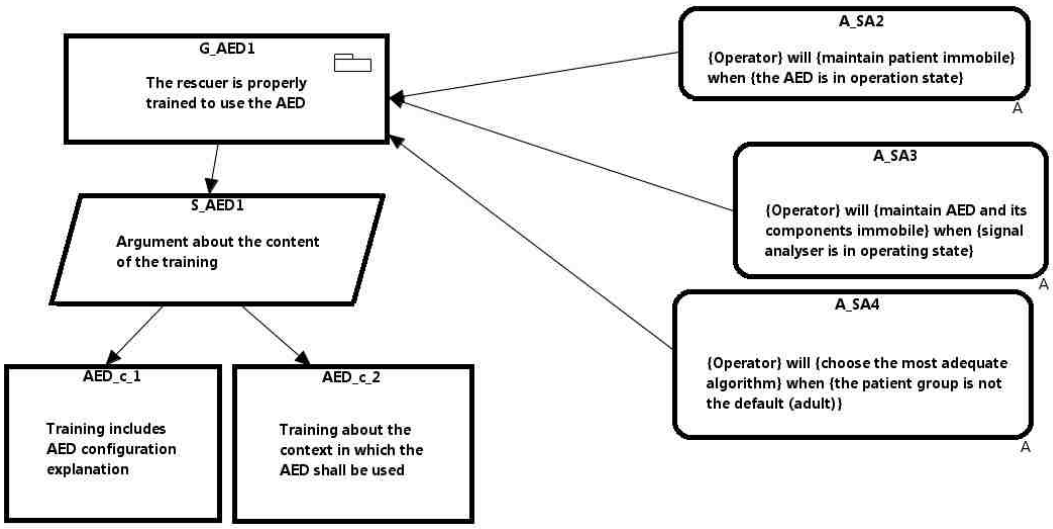


Fig. 114 Excerpt of the AED assurance contract

Measuring Safety Performance by the number of injuries you have is like measuring parenting by the number of smacks you give" – Dr Robert Long

8

Evaluation

Various means of evaluation were employed during the different stages of the research. These include:

- Peer review
- Formalization and tool support
- Case studies

8.1 Scope of the evaluation

In previous chapters the different challenges and proposal have been explained and analysed. In each of the chapters 4, 5 and 6 we have highlight the contributions made in order to fulfil the challenges identified. We can summarize the contributions as follow:

- Common certification Language applied to compositional assurance
  - Extension of SACM metamodel
- Compositional assurance methodology
- Contract-based methodology
- Tool support for the previous concepts

8.2 Case studies

In the previous chapter three different case studies were presented, each of them has been developed in a different domain. We have analysed the automotive, avionics and the medical devices environments with each particularities. We have applied the same approach presented on previous chapters in all the case studies.

For the avionics use case, different interviews have been made with members of the Airworthiness directorate from Thales Avionics Group also different visits to their offices have been made in order to understand the business problem, the data provided and the approach presented.

In the automotive case study, the link has been made across CRF (Centro di Researche the Fiat) as the main contributor for the definition, and with numerous interviews, teleconferences and discussions on the approach and appliance of the proposed methodology.

The medical device use case has been done in collaboration with NUTES. NUTES is part of an initiative for promoting the technological development of Brazil, where the Brazilian

Health Ministry has started some technological transfer projects from well-consolidated manufacturers to institutes of science and technologies in order to retain the know-how of manufacturing medical devices inside the country. In this context, the NUTES project is in charge of receiving and improving methodologies for manufacturing AEDs from the Lifemed and providing new improvements. Lifemed is a Brazilian company that among other products is developing and distributing the AEDs subject of the case study. On this case study also the time difference has been a challenge. A visit to NUTES has been done describing the approach following with multiple teleconferences.

8.2.1 Evaluation strategy

When defining the evaluation strategy for the case studies two main objectives have been identified. The demonstration that the presented approach:

- **Goal 1:** is the composition assurance challenges and
- **Goal 2:** is efficient when handling the composition assurance

In order to measure the two areas mentioned we proposed the creation of an evaluation framework to case studies, which is based on the Goal-Question-Metric (GQM) approach [Solingen Berghout 1999]. In Fig. 115 is shown the approach followed to define the metrics.

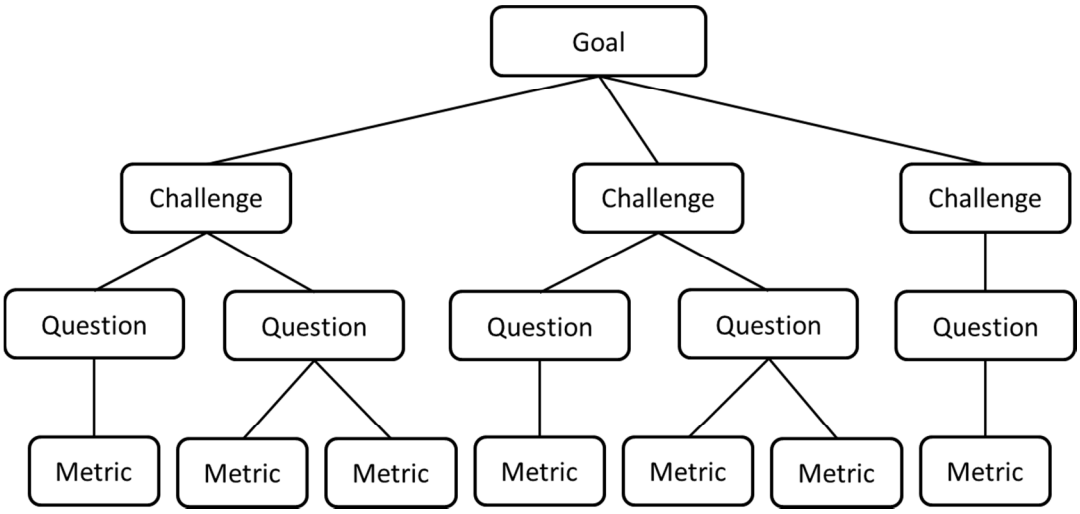


Fig. 115 Goal-Chllenge-Question-Metric approach used

For the **goal 1** about the capability to handle the compositional assurance we have trance the challenges identified and managed along the chapters. The number included in the challenge column is the one used to identify the challenge in the previous chapters as used in tables: Table 9, Table 10 and Table 14. We specified the questions to in a form of true/false statements so only true statement will be considered as acceptable mean to support the approach. Goal 1 measures the capability to answer the research questions **RQ1, RQ2 and RQ3**.

For the **goal 2** focused on the **efficiency**, again the GQM approach has been followed but this time; questions are not related to the challenges but to three areas:

- Systematic
- Transparency
- Methodological

In Table 22 the questions and metrics associated to **goal 1** are specified following the Goal-Challenge-Question-Metric.

Table 22 Metrics specification for goal 1.

Goal	Challenge	Question	Metric	Id
Capability for composition assurance	Standard identification. (4.1)	Can we indicate which standard a component should work to on the baseline?	Number of standards required Number of standards referring on the baselines	1
	Level of compliance (4.2)	Is it feasible to indicate up to what level the component is complying with a standard?	Number of artefact required to comply with the standard Compliance maps existing about those artefacts	2a
			Number of standard requirements Compliance maps about those requirements	2b
	Evidence detail level (4.3)	Are we capable to define which claim a specific piece of evidence is supporting?	Number of evidence not linked on the argumentation	3
	Different formats, not recognizable information (4.4)	Are the standards recognizable by all stakeholders?	Is the reference framework shared by the assurance projects of the case study?	4a
			Are we able to model activities in the same way along the case studies?	4b
			Are we able to model artefacts in the same way along the case studies?	4c
			Are we able to model requirements in the same way along the case studies?	4d

Goal	Challenge	Question	Metric	Id
	Assumptions (4.5)	Is it feasible to handle assumptions of the component?	Number of assumptions defined for the component	5
	Guarantees (4.6)	Is it feasible to handle guarantees of a component?	Number of guarantees defined for the component	6
	Responsibility decompositio n (5.1)	Is it possible to assign complying responsibilities to the different teams?	Is the subproject property used on the case study?	7a
			Are the responsible for each subproject identified on the responsible property?	7b
			Has each of the assurance projects its own baseline indicating the compliance requirements the project will work to?	7c
	Need to reference to others' activities (5.2)	Is it feasible to reference other components or the system development activities?	Are there activities from other projects referenced on the integration subproject?	8
	Reference to others' evidences (5.3)	Is it feasible to reference other components or the system development evidences?	Are there evidences from other projects referenced on the integration subproject?	9
	Evidence refinement by composing artefacts from different components (5.4)	Are we capable to trace evidence composition?	Is any of the evidence from the component part of project evidence?	10a
			Is the artefact part concept being used?	10b
	Need to show integration from the assurance perspective (5.5)	Are we capable to define a way to show the assurance requirements for the integration?	Is there a baseline which includes the integration activities?	11a
			Is there a baseline which includes the integration requirements?	11b

Goal	Challenge	Question	Metric	Id
			Is there a baseline which includes the integration artefacts?	11c
	Capability to trace what each team is doing regarding assurance (5.6)	Are we able to trace each assurance project status?	Does the online compliance report show the status of the assurance projects?	12
	Human factor (6.1)	Are we capable to guide the user on the different topics he/she should treat on the composition?	Do the assumptions or guarantees specified not included in any of the categories?	13
	Validation and checking (6.2)	Do structure formalisation and structure expression support validation and checking?	Are assumptions and guarantees clearly identified?	14
	Interoperability between different suppliers (6.3)	Are we able to express guarantees and assumptions on a uniform way?	Are the assumptions on the integration project linked with the component guarantees?	15
	Facilitate the integration of the components within the system (6.4)	Does the contract include enough information to validate the assumptions and guarantees?	Are the assumptions and the guarantees linked of the same category?	16
	Learning curve (6.5)	Are the set of structured expressions too complex to be used?	Number of times a structured expression has been used	17

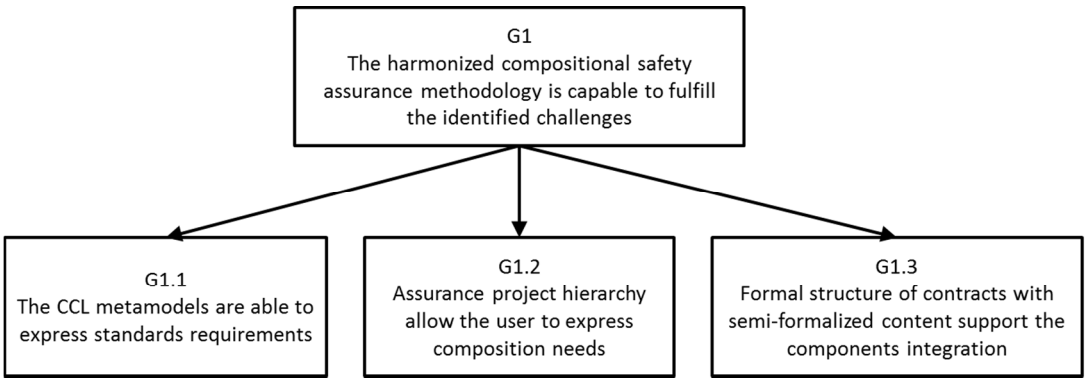


Fig. 116 Excerpt of the argument justifying the capability of the assurance composition approach to fulfil the challenges identified

The goal 1 is decomposed by the sub-goals which are the contributions proposed in this work.

For the goal 2, the questions are not related to challenged but about the provided support.

Goal	Question	Metric	Id
Efficiency on the composition assurance	Is there support for detecting assumptions?	Number of assumptions identified Total number of assumptions	18
	Are the component evidences reusable?	Number of evidences requested on the integration baseline before the component is integrated Number of evidences from the component that map those requirements	19
	Is the scope of the approach sufficiently complex?	Number of compliance activity covered	20 a
		Number of compliance artefacts covered	20 b

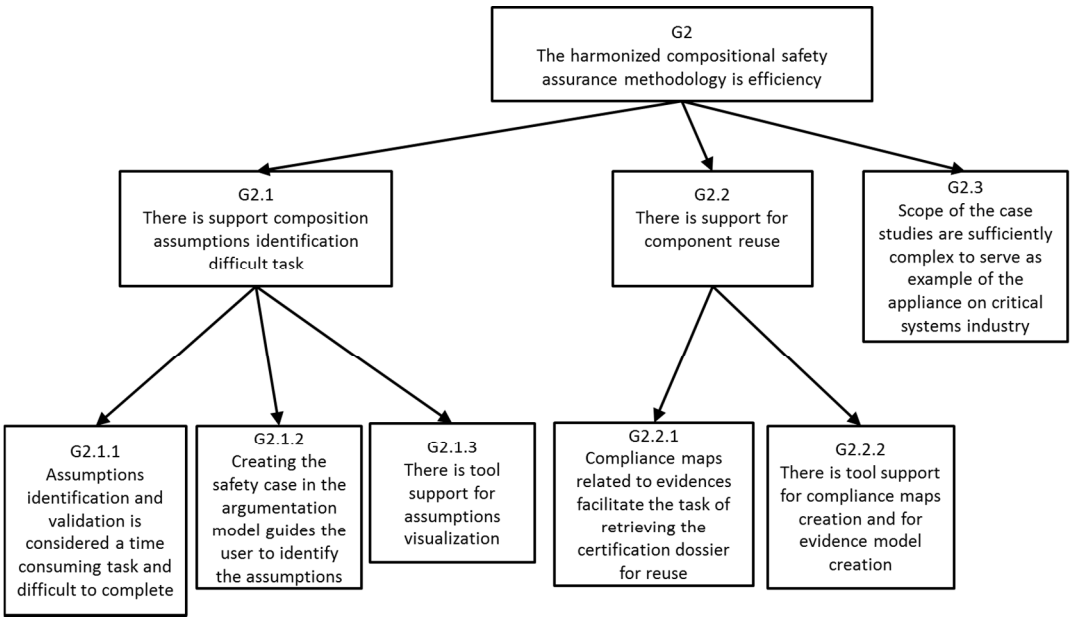


Fig. 117 Argument to support the efficiency metrics definition

The goal 2 is decomposed into sub-goals that will cover parts of the tool support developed and used.

Metric 1

The objective of the metric is to evaluate if the standard modelling approach presented on chapter 4 let the user specify which standard the component is complying. The use of baselines was the mechanism proposed to reference at the component assurance project level which standards the component was aiming to comply with.

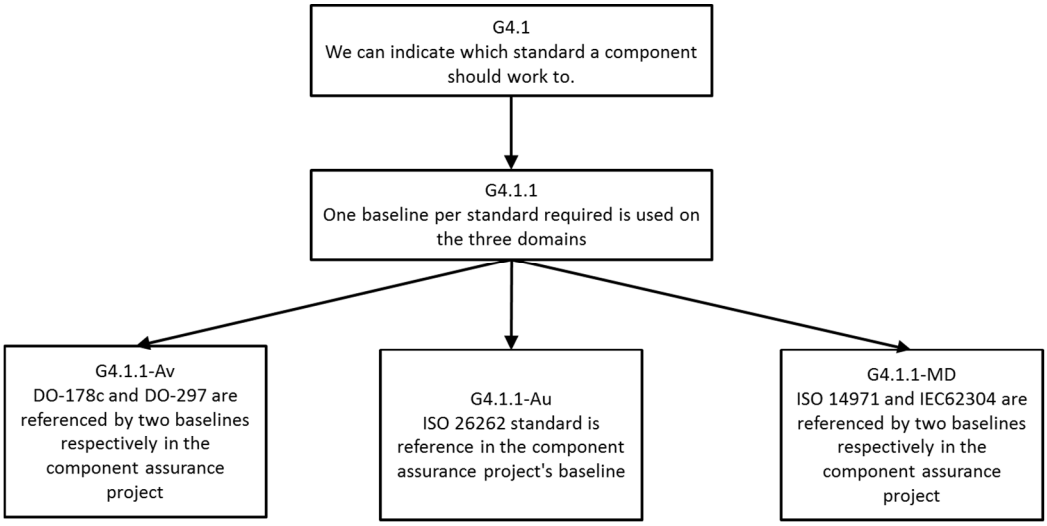


Fig. 118 Argument justifying metric1 objective

Metrics 2a and 2b

The aim of these metrics is to ensure that the compliance map mechanism defined on chapter 4 is valid. The mechanism is appropriate for indicating the level of compliance of a component has regarding a standard. With respect to metric 2a the focus is on the artefacts used as compliance evidence regarding documentation requested by the standard. If the artefacts required by the standard are covered by the compliance maps, then the objective is fulfilled. Metric 2b is similar but the focus is not on the required artefact but on the requirements to fulfil. Compliance maps also show the trace between the requirements and the claim related on the argumentation where it is being decomposed and supported.

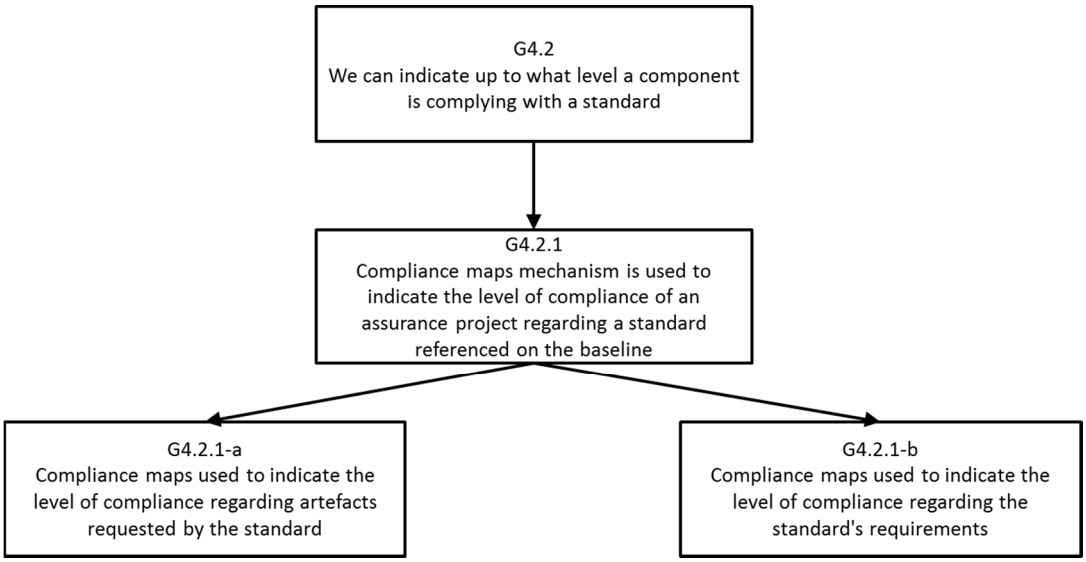


Fig. 119 Argument justifying metric 2a and 2b objectives

Metric 3

This metric tries to evaluate where we have been able to trace how the evidence is supporting a claim. The objective is to have a no answers or at least a minimum value, meaning that all evidence is supporting a claim and it is traceable. If there is evidence not linked with any argumentation, then we are not able to affirm that we are capable to make this trace in all scenarios.

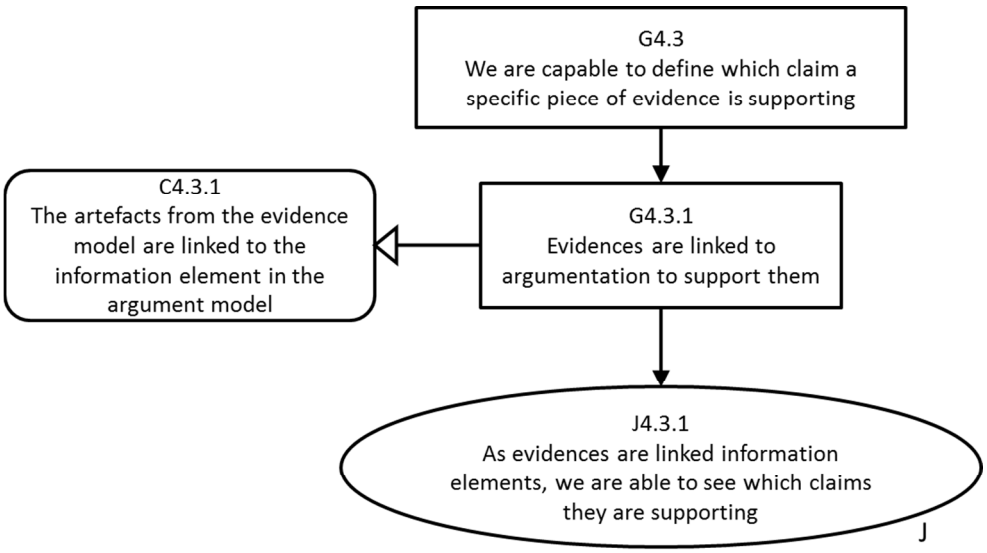


Fig. 120 Argument justifying metric3 objective.

Metric 4a, 4b, 4c, 4d

All these metrics expect "yes or no" answer. The metric 4a aims to detect whether the different stakeholders are able to share the information about the standard. If they are able to share the same reference frameworks between the different assurance cases, means that the information is understandable.

The metrics 4b, 4c and 4d focus on the capability to model the main elements we can identify on the standards.

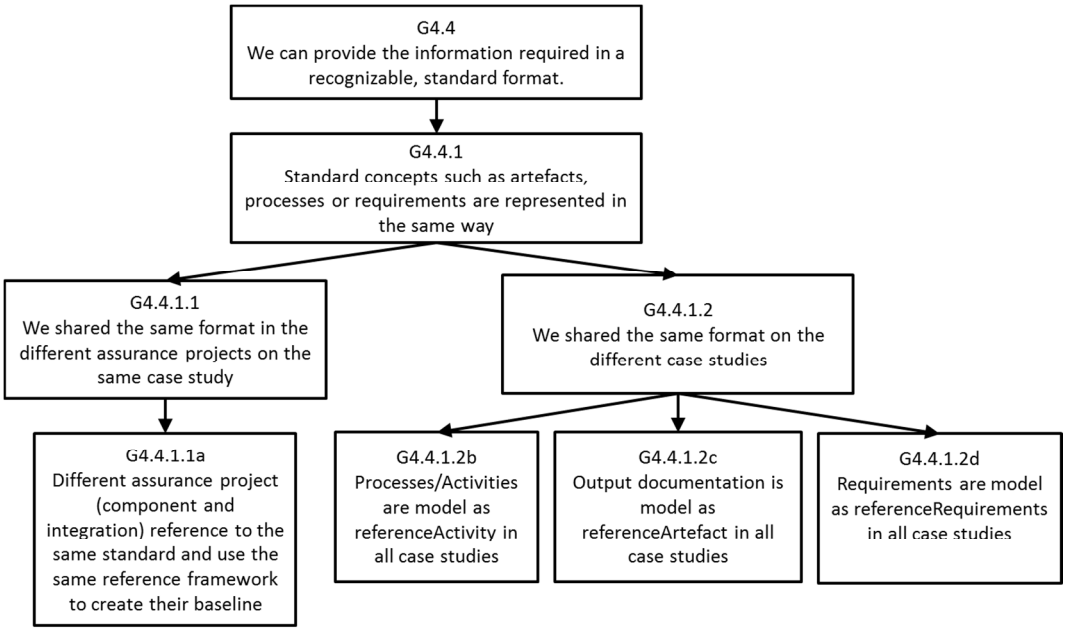


Fig. 121 Argument justifying metric 4a,4b,4c and 4d objectives

Metric 5

In all component development we need to make assumptions. The nature of the component is to be reused and the context/objective for use… might change. The assumptions define the set of properties/activities/artefacts the component developer expect the integrator to take care about. The objective of this metric is to check the capability to define the assumptions. If the number is less than one, it means that no assumptions has been done and will indicate that we are not able to claim the approach let the user define assumptions.

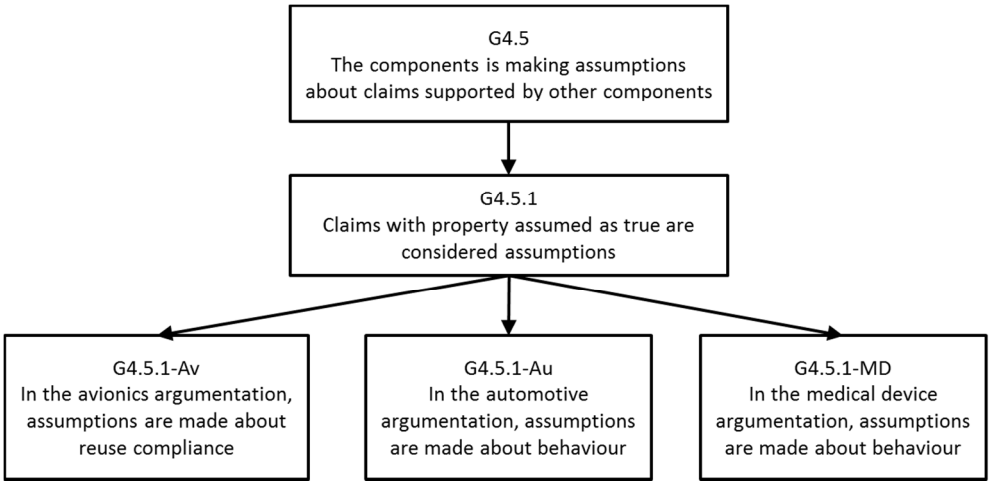


Fig. 122 Argument justifying metric5 objective

Metric 6

This metric similar to the previous one but the focus is on the guarantees that a component provides to the rest. If the number is less than one, it means that no guarantees has been done and will indicate that we are not able to claim the approach let the user define guarantees.

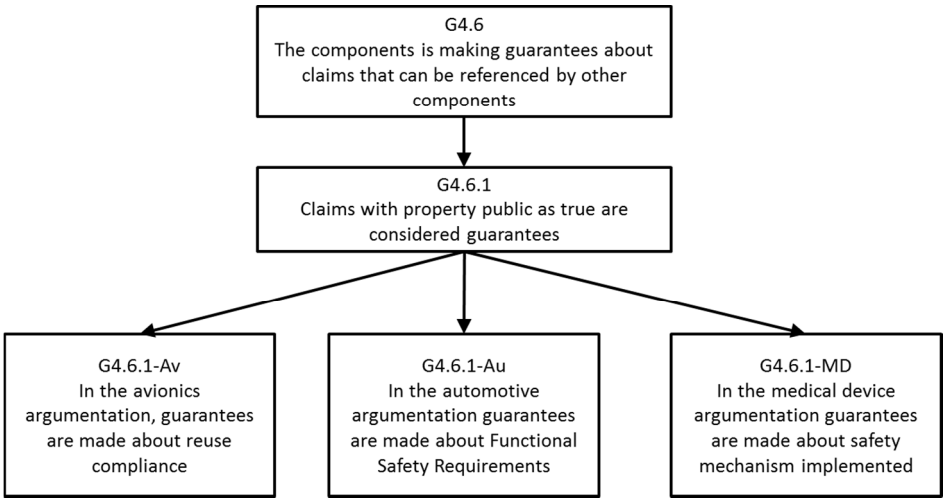


Fig. 123 Argument justifying metric6 objective.

Metrics 7a, 7b and 7c

All these metrics expect “yes or no” answer. The metric 7a focused on the hierarchy approach where an assurance project can be a subproject from another assurance project. Metric 7b highlights the responsible person in charge of the project assurance. Metric 7c indicates the responsibility in the way as the assurance requirements the assurance project has planned to comply with.

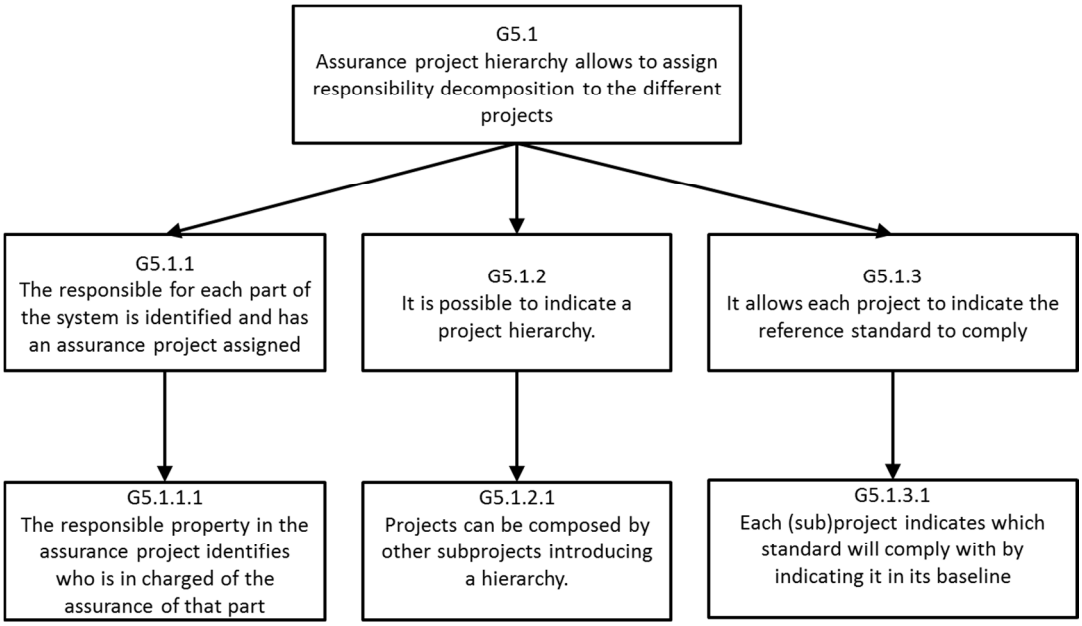


Fig. 124 Argument justifying metric 7a, 7b and 7c objectives.

Metric 8

The aim of this metric is to ensure that it is feasible to references activities among the subprojects. It is a yes or no question and yes is the expected answer.

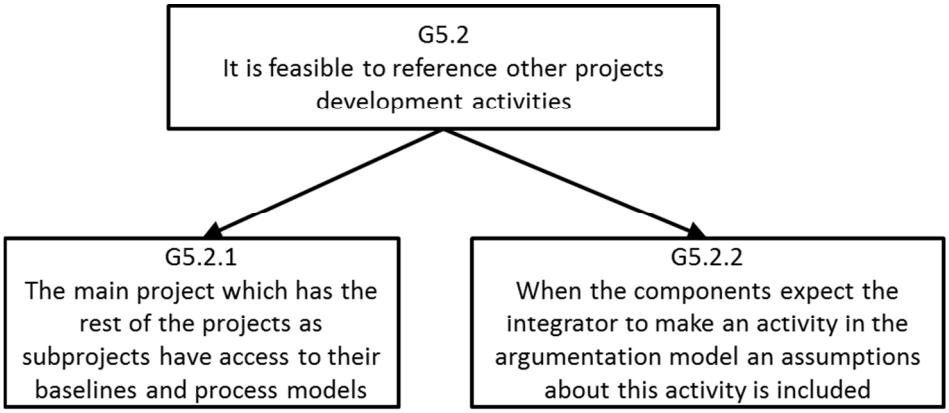


Fig. 125 Argument justifying metric8 objective.

Metric 9

The aim of this metric is to ensure that it is feasible to references evidences among the subprojects. It is a yes or no question and yes is the expected answer.

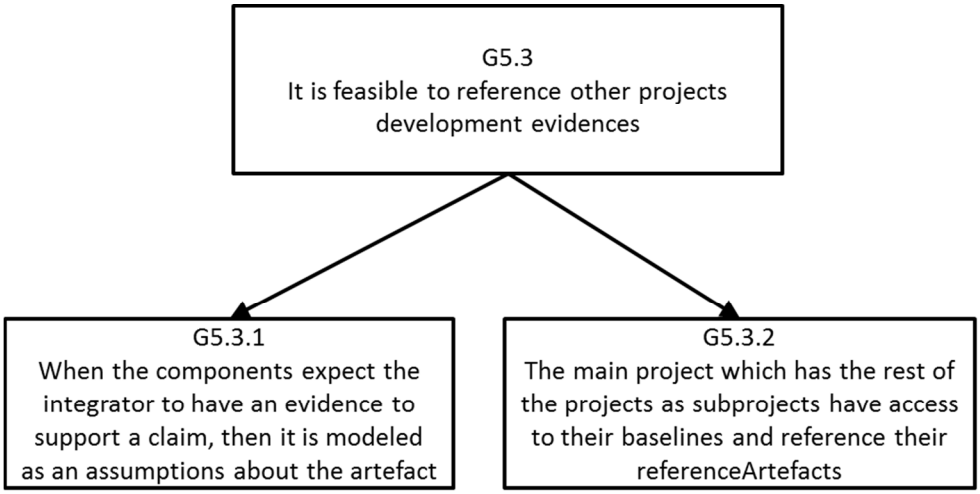


Fig. 126 Argument justifying metric9 objective.

Metrics 10a and 10b

These two metrics has the objective to show that composition of evidences is feasible, they are related to the previous one. First metrics deals with referring a piece of evidence from another subproject (A) as part of the subproject's (B) evidences. The second metric is about the use of artefact part concept defined on the evidence metamodel from chapter 4. This is the mechanism proposed to indicate that an artefact (a document, a section, a file, some code) is part of a bigger artefact.

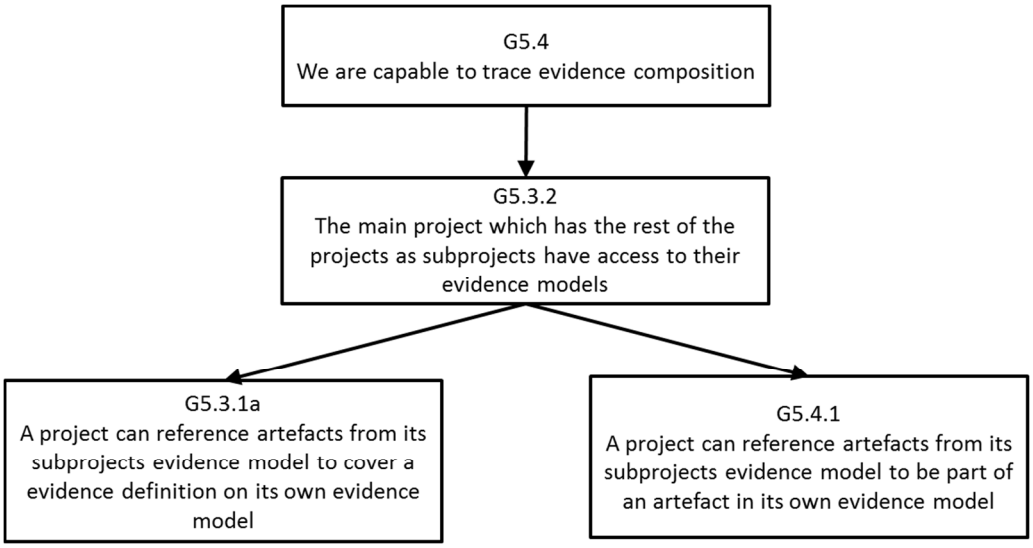


Fig. 127 Argument justifying metric 10a and 10b objectives

Metric 11a, 11b and 11c

These metrics focus on the capability to show the integration requirements from the assurance perspective. Assurance request to execute integration activities mentioned on the standard and that is the target for the first metrics. The activities should be performed in order to fulfil certain requirements which are the objective of the next metric and finally as a result some artefacts should be the outputs which are question about on the last metric.

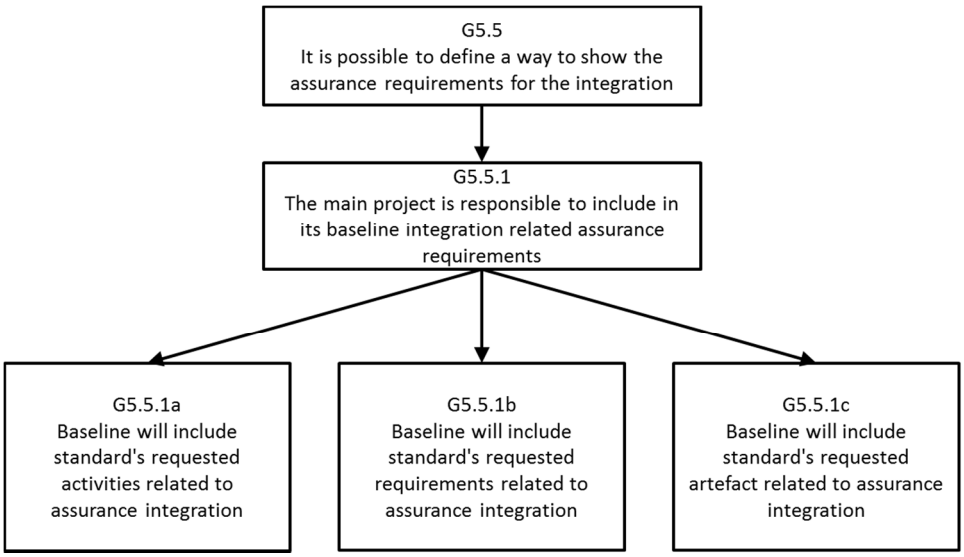


Fig. 128 Argument justifying metric 11a, 11b and 11c objectives

Metric 12

This metric questions about the capability of the online compliance report to show the status of the assurance projects. This metric is not an objective metric because users might differ on the idea of what concepts should appear on a report. However, the objective here is to show a quick view of the status.

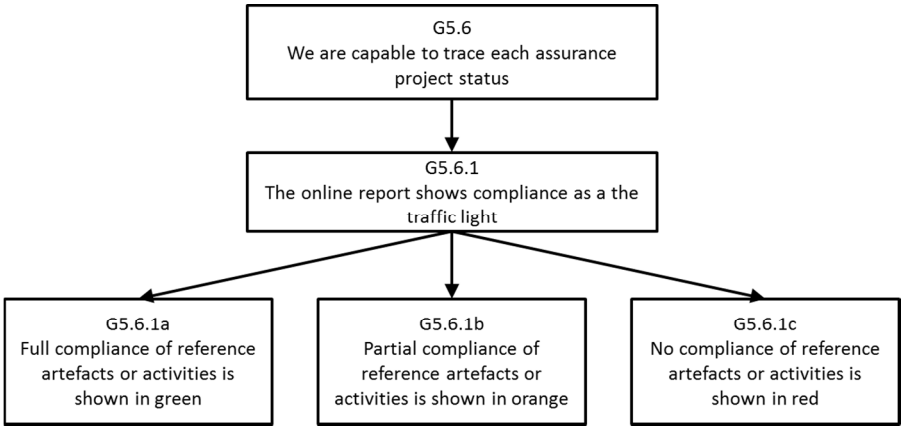


Fig. 129 Argument justifying metric 12 objective.

Metric 13

This metric questions the completeness of the categories list. The objective with the category classification is to classify all types of information that will be handled on the contracts. If there are too many assumptions or guarantees with no category associated, then the categories classification might not be complete.

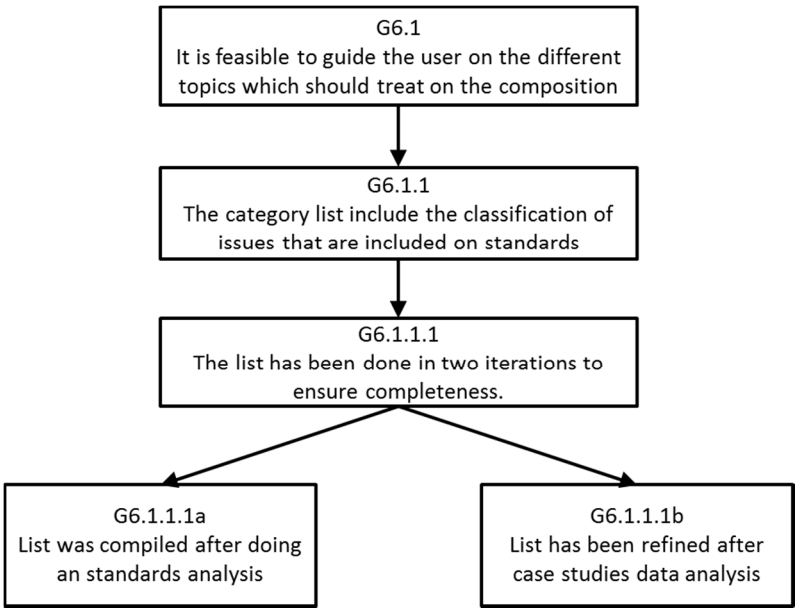


Fig. 130 Argument justifying metric13 objective

Metric 14

This metric focus on the support for contract checking and validation. In order to confirm that this is feasible we inquire about the possibility for tool support. We understand that if there is a tool support then there is possibility for validation development.

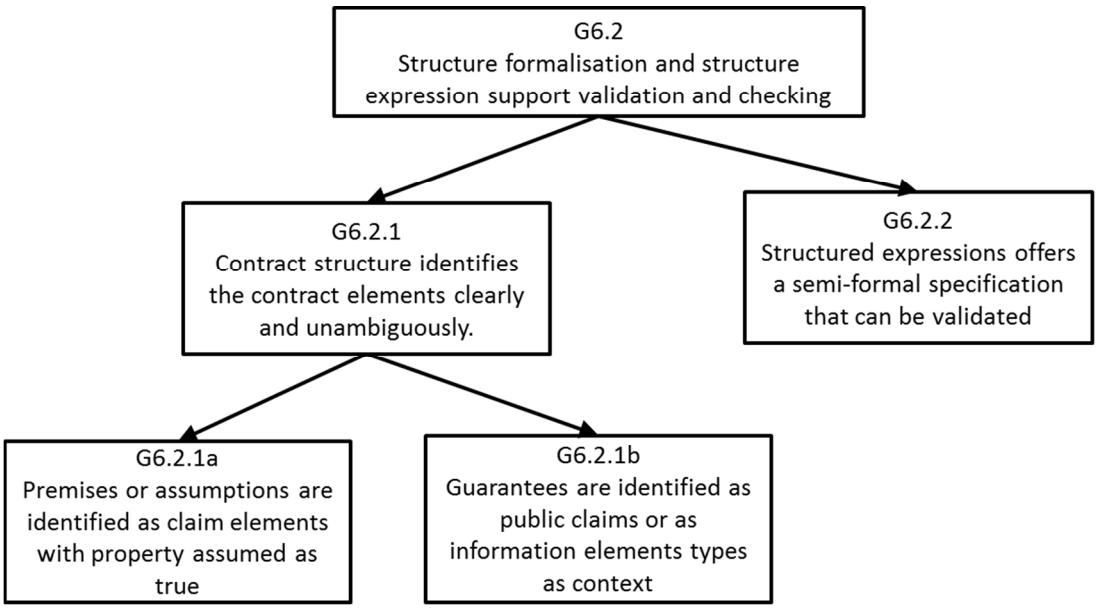


Fig. 131 Argument justifying metric14 objective

Metric 15

This metric focus on the level of understanding between different suppliers. On metrics 5 and 6 we have focused on the capability to specify assumptions and guarantees. This metric focused whether on the case study it has been possible to link the functional behaviour requested on the integration to the guarantees offered by the component.

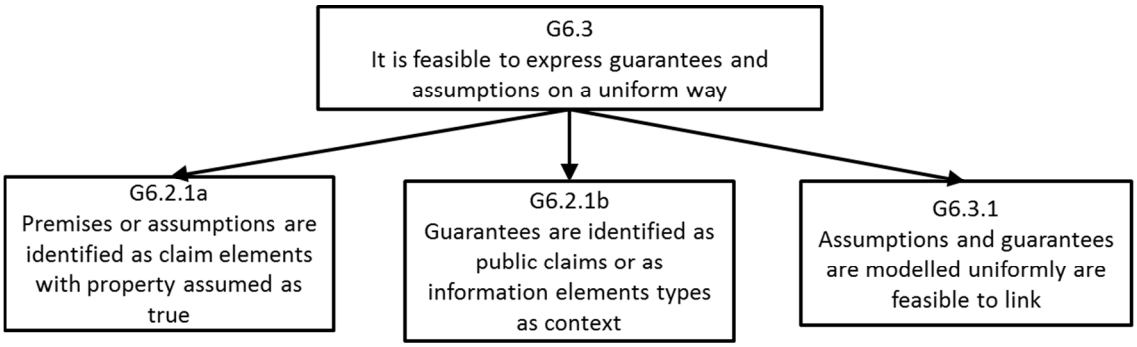


Fig. 132Argument justifying metric15 objective

Metric 16

The objective of this metric is to inquire about the contracts. Categories have been design in order to support the user when validating the assumptions with the guarantee information. The idea is to link information of the same category. If there are assumptions linked with guarantees of a different category, the hypothesis of linking information on the same kind will no longer be true.

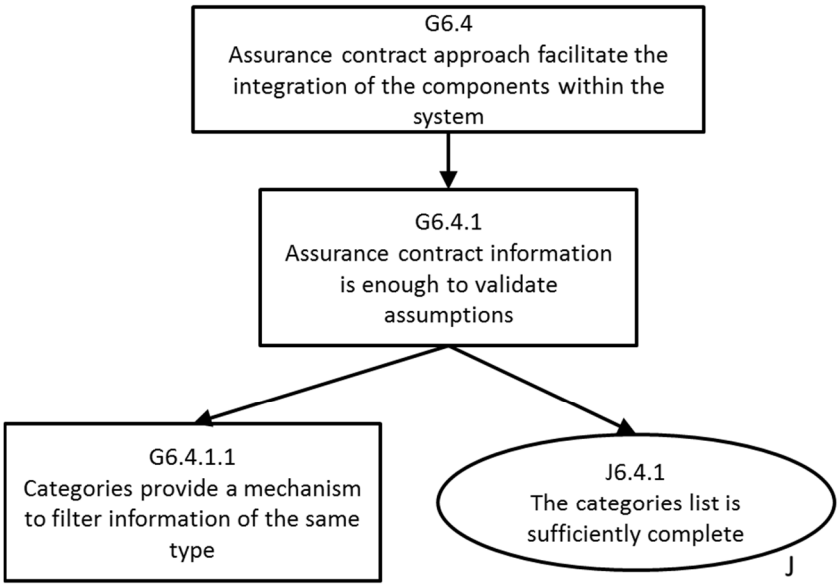


Fig. 133 Argument justifying metric16 objective

Metric 17

The objective of this metric is to ensure the reusability. The idea is not to have so many structured expressions that will end up with an encyclopaedia of expressions. The objective is to identify those expressions which are highly reusable. If an expression is often used, then it indicates that we have achieved on our goal.

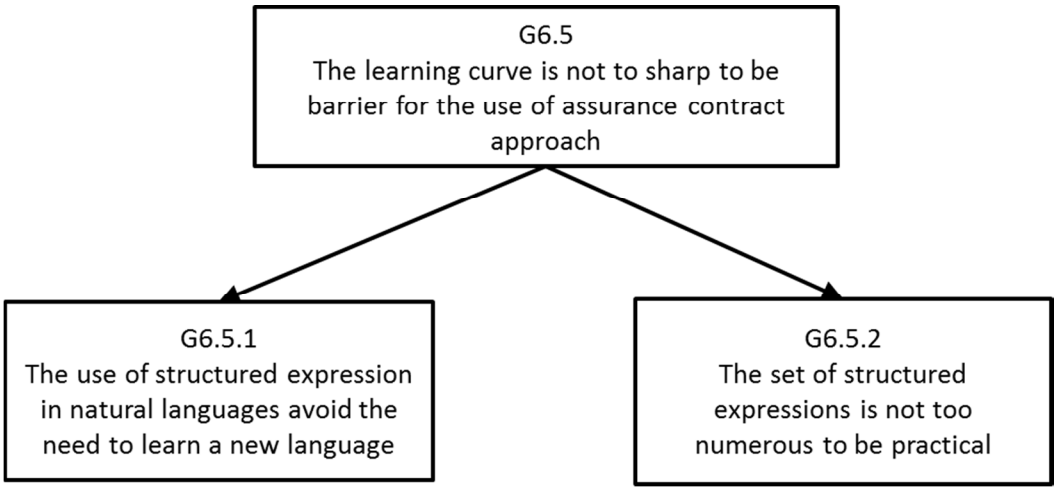


Fig. 134 Argument justifying metric17 objective

Metric 18

This metrics is focused on efficiency and efforts saving. Usually the task of identifying assumptions is time consuming. If we assume that we need a specific amount of time for each of the assumptions to be identified, this time will be reduced if the assumptions are

already detected. The metric calculates the number of assumptions detected against the total number of assumptions so as to calculate the percentage saved.

Metric 19

This metric is very similar to the previous one. Collecting evidences is also a time consuming task. The metric focused on calculating the percentage of evidences requested that are already included on the component assurance project. The percentage obtained will be translated into efficiency.

Metrics 20a and 20b

These metrics are focusing on the scope of the approach. It focused on the complexity of the case studies in order to provide confidence to the rest of metrics. If the number is too low it will be consider a too simplistic case study, while a high number will give us a view of the complexity of the work

8.2.2 Avionics

Similar to the previous one, this case study has gone through an evaluation phase after the approach has been followed in order to get all the measures.

The objective of this case study was the composition of an IMA software module within an IMA platform.

In Table 23 the values of the metrics extracted from the automotive case study are shown.

Table 23 Metric values obtained in the avionics case study

Id	Metric	Value	Comments
1	Number of standards required	2	Standards identified are DO-297 and DO 178c
	Number of standards referring on the baselines	2	
2a	Number of artefact required to comply with the standard	24+16	For the IMA module development (artefacts requested from DO-178C and DO-297)
	Compliance maps existing about those artefacts	24+16	
2b	Number of standard requirements	527	For the IMA module development (requirements requested from DO-178C and DO-297)
	Compliance maps about those requirements	527	
3	Number of evidence not linked on the argumentation	0	
4a	Is the reference framework shared by the assurance projects of the case study?	YES	

Id	Metric	Value	Comments
4b	Are we able to model activities in the same way along the case studies?	YES	
4c	Are we able to model artefacts in the same way along the case studies?	YES	
4d	Are we able to model requirements in the same way along the case studies?	YES	
5	Number of assumptions defined for the component	7	The ARINC 653 already defines the reference architecture
6	Number of guarantees defined for the component	27	
7a	Is the subproject property used on the case study?	YES	
7b	Are the responsible for each subproject identified on the responsible property?	YES	
7c	Has each of the assurance projects its own baseline indicating the compliance requirements the project will work to?	YES	AT component there are 2 baselines (per standard), at integration 2 baselines (one to reference the module and another one for the integration)
8	Are there activities from other projects referenced on the integration subproject?	YES	
9	Are there evidences from other projects referenced on the integration subproject?	YES	
10a	Is any of the evidence from the component part of project evidence?	YES	MAP is requested on the MAAS
10b	Is the artefact part concept being used?	YES	
11a	Is there a baseline which includes the integration activities?	YES	
11b	Is there a baseline which includes the integration requirements?	YES	

Id	Metric	Value	Comments
11c	Is there a baseline which includes the integration artefacts?	YES	
12	Does the online compliance report show the status of the assurance projects?	YES	
13	Do the assumptions or guarantees specified not included in any of the categories?	NO	
14	Are assumptions and guarantees clearly identified?	YES	
15	Are the assumptions on the integration project linked with the component guarantees?	YES	
16	Are the assumptions and the guarantees linked of the same category?	YES	
17	Number of times a structured expression has been used	Min:1 Max:9	Activity Adequacy Claims Type 4: 6 Activity-Artefact Claim Type 1: 6 Artefact Compliance Claims Type 2: 9 Element adequacy claims Type 1: 2 Element behaviour claims Type 1: 1 Element compliance claims Type 1: 7 Fault Accommodation Claims Type 5: 1 Hazard Mitigation Claims Type 3: 2
18	Number of assumptions identified	7	
	Total number of assumptions	7	
19	Number of evidences requested on the integration baseline before the component is integrated	16	Just the ones requested by DO-297
	Number of evidences from the component that map those requirements	16	
20a	Number of compliance activity covered	53+15	DO-178C+DO-297
20b	Number of compliance artefacts covered	24+24	DO-178C+DO-298

This case study has mainly been focused on compliance. The safety case for the IMA module includes arguments about:

- Auto generated argumentation in relation with DO-178C and DO-297 standards conformity
- Completeness of verification artefacts
- Segregation insurance between partitions

This has a direct effect on the assumptions and guarantees as the categories used are mainly about activities, artefacts and elements compliance.

In this use case, artefacts are linked to constrained requirements in a sense that describe the information that should contain and the process followed to produce them. That is the rationale behind such a high number of requirements.

In the avionics domain, the ARINC 653 Avionics Application Standard Software Interface already defines most of the assumptions that appear on software development, which explains the reason why the number of assumptions is so low.

One of the big challenges of this use case has been that data has been sanitized before it was used for the case study. That has implied some iterations and meetings with people from Thales certification directorate in order to ensure the understanding of the data provide and request some more data to reduce the effects on the sanitization. On the other hand, usage domain rules that have been used to provide the guarantees and assumptions are already defined on a formal way. For that reason the effect of the structure language expressions used has not been too much noticed.

8.2.3. Automotive

Once the case study has followed the approach described on this thesis, the case study has gone through an evaluation phase where all the values of the metrics have been extracted.

The objective of this case study was the composition of a system SEooC with an electric parking functionality within a full electric vehicle.

In the following table the values of the metrics extracted from the automotive case study are shown.

Table 24 Metric values obtained in the automotive case study

Id	Metric	Value	Comments
1	Number of standards required	1	ISO 26262
	Number of standards referring on the baselines	1	
2a	Number of artefact required to comply with the standard	25	For the SEooC development. Some of the documents are supporting

Id	Metric	Value	Comments
	Compliance maps existing about those artefacts	21	process and are optional, those are the ones that has missing compliance maps
2b	Number of standard requirements	34	For the SEooC development.
	Compliance maps about those requirements	34	
3	Number of evidence not linked on the argumentation	0	
4a	Is the reference framework shared by the assurance projects of the case study?	YES	
4b	Are we able to model activities in the same way along the case studies?	YES	
4c	Are we able to model artefacts in the same way along the case studies?	YES	
4d	Are we able to model requirements in the same way along the case studies?	YES	
5	Number of assumptions defined for the component	13	
6	Number of guarantees defined for the component	23	
7a	Is the subproject property used on the case study?	YES	
7b	Are the responsible for each subproject identified on the responsible property?	YES	
7c	Has each of the assurance projects its own baseline indicating the compliance requirements the project will work to?	YES	
8	Are there activities from other projects referenced on the integration subproject?	YES	
9	Are there evidences from other projects referenced on the integration subproject?	YES	

Id	Metric	Value	Comments
10a	Is any of the evidence from the component part of project evidence?	YES	On the standard they are mentioned as refinement
10b	Is the artefact part concept being used?	YES	One example is the HARA
11a	Is there a baseline which includes the integration activities?	YES	
11b	Is there a baseline which includes the integration requirements?	YES	
11c	Is there a baseline which includes the integration artefacts?	YES	
12	Does the online compliance report show the status of the assurance projects?	YES	As it is evaluated once the approach has already been done on the total scope, we only see the final status.
13	Do the assumptions or guarantees specified not included in any of the categories?	NO	
14	Is there tool support for contract creation?	YES	
15	Are the assumptions on functional behaviour made at integration level linked with the component guarantees?	YES	Mainly about functional safety behaviour
16	Are the assumptions and the guarantees linked of the same category?	YES	

Id	Metric	Value	Comments
17	Number of times a structured expression has been used	Min:1 Max: 12	Agent Action Claims type 1: 3 Element adequacy claims Type 1: 12 Element adequacy claims type 4: 1 Element behaviour claims type 1: 1 Element behaviour claims type 3: 3 Element behaviour claims type 4: 4 Element behaviour claims type 5: 4 Fault accommodation claims type 1: 1 Fault accommodation claims type 2: 5 Fault accommodation claims type 3: 3 Fault accommodation claims type 4: 1
18	Number of assumptions that were detected	26	Assumptions are focused on technical safety requirements.
	Total number of assumptions	26	
19	Number of evidences requested on the integration baseline before the component is integrated	22	
	Number of evidences from the component that map those requirements	22	
20a	Number of compliance activity covered	30	For the SEooC assurance project
20b	Number of compliance artefacts covered	24	For the SEooC assurance project

This case study has mainly been focused on hazards avoidance. The safety case for the SEooC includes two types of arguments:

- Auto generated argumentation in relation with ISO 26262 conformity
- Hazards identified, and safety requirements derived.

This has a direct effect on the assumptions and guarantees as the categories used are mainly about the behaviour and fault accommodation.

Regarding the evidence, on this use case the different refinements of evidence has been modelled as new evidence. This decision has been taken in accordance with the standard. On the standard each refinement is mentioned as a new document and so, on the modelled each refinement mentioned on the standard is identified as a new piece of

evidence. At the same time each of these refinements has also been modelled with a relationship with the previous one. As it is mentioned, artefacts included on the standard and which are not included on the use case are the ones considered optional.

Audits and verification of critical artefacts such as HARA have also been part of the integration assurance project.

When an activity shall be performed as a succession of tasks, according to the standard, this has been modelled as sub-activities. This is one of the reasons the number of requirements is not very high.

One of the big challenges of this use case has been that there was no data from previous projects that comply with ISO 26262 as it is a quite new standard and only new developments shall follow it. Other important challenge has been the Intellectual Property for the documents used as evidence. They have followed a sanitized process and only information relevant for the case study was available. Fortunately this has been resolved by a fluent communication with CRF staff, mainly Alberto Melzi.

8.2.4 Medical device

This has been the last case study implemented. The objective of this case study was the composition of the signal analyser software into an AED. This is a critical component. The development of the component has been done in parallel of recollecting the data for implementing the presented approach.

In Table 25 the values of the metrics extracted from the automotive case study are shown.

Table 25 Metric values obtained in the medical device case study

Id	Metric	Value	Comments
1	Number of standards required	2	IEC 62304 only software development process scope and ISO 14971
	Number of standards referring on the baselines	2	
2a	Number of artefact required to comply with the standard	15+16	For the IEC 62304 and for ISO 14971 respectively for the signal Analyser component
	Compliance maps existing about those artefacts	15+16	
2b	Number of standard requirements	40	
	Compliance maps about those requirements		
3	Number of evidence not linked on the argumentation	0	
4a	Is the reference framework shared by the assurance projects of the case study?	YES	

Id	Metric	Value	Comments
4b	Are we able to model activities in the same way along the case studies?	YES	
4c	Are we able to model artefacts in the same way along the case studies?	YES	
4d	Are we able to model requirements in the same way along the case studies?	YES	
5	Number of assumptions defined for the component	6	
6	Number of guarantees defined for the component	30	
7a	Is the subproject property used on the case study?	YES	
7b	Are the responsible for each subproject identified on the responsible property?	YES	
7c	Has each of the assurance projects its own baseline indicating the compliance requirements the project will work to?	YES	
8	Are there activities from other projects referenced on the integration subproject?	YES	
9	Are there evidences from other projects referenced on the integration subproject?	YES	
10a	Is any of the evidence from the component part of project evidence?	YES	
10b	Is the artefact part concept being used?	YES	
11a	Is there a baseline which includes the integration activities?	YES	
11b	Is there a baseline which includes the integration requirements?	YES	
11c	Is there a baseline which includes the integration artefacts?	YES	
12	Does the online compliance report show the status of the assurance projects?	YES	
13	Do the assumptions or guarantees specified not included in any of the categories?	NO	
14	Are assumptions and guarantees clearly identified?	YES	

Id	Metric	Value	Comments
15	Are the assumptions on the integration project linked with the component guarantees?	YES	
16	Are the assumptions and the guarantees linked of the same category?	YES	
17	Number of times a structured expression has been used	Min:1 Max:7	Agent Action Type 1: 5 Agent Action Type 1: 6 Component Development Type 4: 1 Element behaviour Type 1: 5 Element behaviour Type 3: 7 Element behaviour Type 4: 2 Element behaviour Type 5: 7 Fault accommodation Type 3: 5 Fault accommodation Type 4: 1
18	Number of assumptions identified Total number of assumptions	7	
19	Number of evidences requested on the integration baseline before the component is integrated Number of evidences from the component that map those requirements	12	This evidences from the component are part of the AED evidences
20 a	Number of compliance activity covered	11+7	ISO 14971+IEC62304
20 b	Number of compliance artefacts covered	16+12	ISO 14971+IEC62304

The product under analysis in this case study has been designed at the same type as the case study was developed. The product was not created before so the data was created when it was required by applying the approach. This has produced new evidences that were specifically created and also affected the way process of development. It has given us the chance to see the appliance of the approach from the beginning of the system planning.

The Automated External Defibrillator system was designed as a component-based system; however, the components were developed by the same team.

The medical devices safety standards are not prescriptive in terms of evidence to produce for compliance and following this approach has given to the user a mechanism to show exactly the way they were complying with the requirements. Assurance cases are not very extended in this domain; however, they have been much appreciated.

8.2.5 Case studies conclusions

Two goals were tested in the case studies, the capability to handle the assurance challenges and the efficiency when handling composition assurance. The metric and goals are related to the research questions presented in chapter 1.

In Fig. 135 the metrics associated with RQ1 (How can we express standards compliance needs in relation to component development and component integration?) are shown.

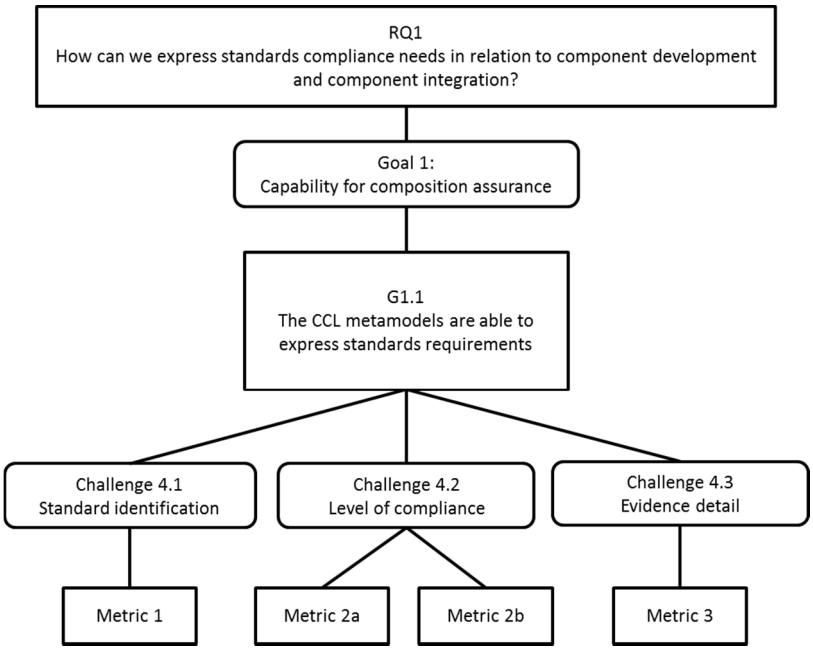


Fig. 135 Metrics associated with RQ1

The all the cases we have been able to model the different standards, in total 5 standards have been modelled. In the case studies not all the requirements for the standards were applied.

In this case we can affirm that the assurance modelling contribution proposed is a valid method for expressing standards compliance needs in relation to component development and component integration

In Fig. 136 the metrics associated with RQ2 (How can we express the distribution of responsibility across the system stakeholders in relation to safety standards compliance?) are shown.

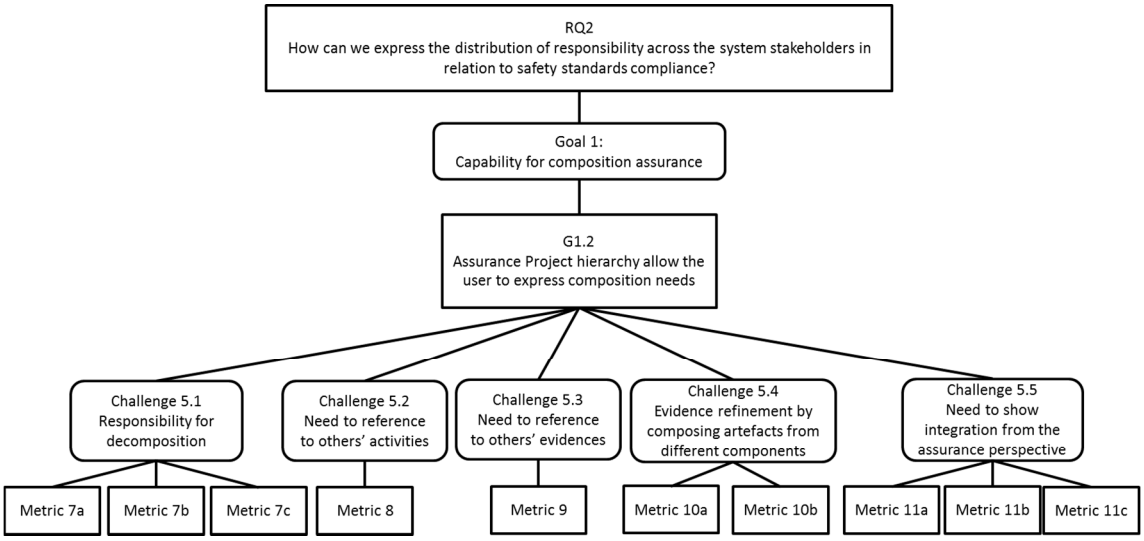


Fig. 136 Metrics associated with RQ2

The compositional assurance approach validated has not only made an impact in the assurance process but also indirectly in the development processes, making component responsible aware of their responsibilities and the needs from the system perspective.

The metrics referring to RQ3 (How can we best support stakeholders while ensuring compliance across the system development lifecycle?) are shown in Fig. 137.

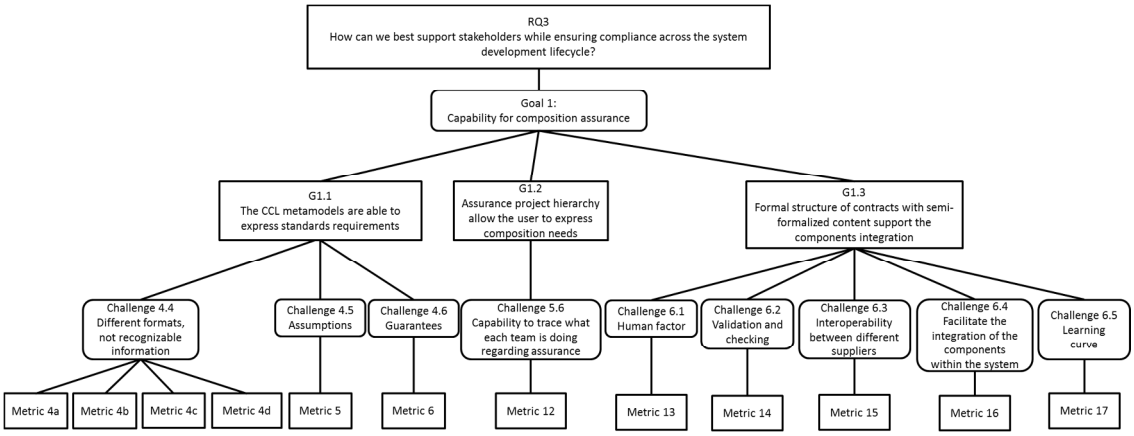


Fig. 137 Metrics associated with RQ3

RQ3 is the one with more metrics associated and it is also associated with goal 2, being efficient when handling the composition assurance. The tool has been able to support the process and the structured expressions have support not only the specification but also at design time, it has been beneficial to detect assumptions that have not been initially identified. However the results for this question might be the most subjective.

8.3. Evaluation through peer review

The research outcomes of this thesis were presented to, and reviewed by, various members of the academia and systems and functional safety engineers in industry. The thesis contributions have taken advantage of peer review in order to assure the adequacy of the other, more formal, mean of evaluation (outlined in the previous two sections) as well as to mitigate against any limitations of case studies.

The following projects in which the author of the thesis has collaborated and in some of them she continues collaborating have served to enhance the conceptual work of the thesis.

8.3.1 Recomp Project

"RECOMP" stands for Reduced Certification Costs Using Trusted Multi-core Platforms and is a European funded project from ARTEMIS JOINT UNDERTAKING (JU). The project started April 1th of 2010 and has duration of 36 months.

RECOMP research project pretended to form a joint European task force contributing to the European Standard Reference Technology Platform for enabling cost-efficient certification and re-certification of safety-critical systems and mixed-criticality systems, i.e. systems containing safety-critical and non-safety-critical components. The aim was to establish methods, tools and platforms for enabling cost-efficient (re-)certification of safety-critical and mixed-criticality systems. Applications addressed were automotive, aerospace, industrial control systems, and lifts and transportation systems.

This project has been the first step of the thesis and where the analysis of the standards from the composition point of view started.

In this project the author of the thesis worked on:

- Analysis of the DO 178 and IEC 61508 standards regarding reuse requirements
- Modular safety cases approach to be used for safety assessment
- Support to the tool chain identification to reduce certification costs on multicores
- Appliance of modular safety cases on an avionics demonstrator.

8.3.2 OPENCOSS

The work presented here was framed under collaboration within the European project called OPENCOSS (Open Platform for EvolutioNary Certification Of Safety-critical Systems) which is a large-scale collaborative project of the EU's Seventh Framework Program. [OPENCOSS]

OPENCOSS is a European large scale integrating FP7 project dedicated to produce the first European-wide open safety certification platform: an Open Platform for EvolutioNary Certification Of Safety-critical Systems for the railway, avionics and automotive markets.

Most of the work presented on this thesis has been framed under the collaboration within this project.

In this project the author of the thesis worked on:

- Developing the platform architecture
- Support on creation of the common certification language, specially the argumentation metamodel as an extension of the SAM model and identification the links between the different meta-model.
- Research on the compositional certification area
- Support on the case studies from the avionics and the automotive domains

8.3.3 SafeAdapt

The promising advent of fully electric vehicles also means a shift towards fully electrical control of the existing and new vehicle functions. In particular, critical X-by-wire functions require sophisticated redundancy solutions. As a result, the overall Electric/Electronic (E/E) architecture of a vehicle is becoming even more complex and costly.

The main idea of SafeAdapt is to develop novel architecture concepts based on adaptation to address the needs of a new E/E architecture for FEVs regarding safety, reliability and cost-efficiency. This will reduce the complexity of the system and the interactions by generic, system-wide fault and adaptation handling. It also enables extended reliability despite failures, improvements of active safety, and optimized resources. This is especially important for increasing reliability and efficiency regarding energy consumption, costs and design simplicity.

SafeAdapt follows a holistic approach for building adaptable systems in safety-critical environments that comprises methods, tools, and building blocks for safe adaptation. This also includes certification support of safety-critical systems in the e-vehicle domain. The technical approach builds on a SafeAdapt Platform Core, encapsulating the basic adaptation mechanisms for re-allocating and updating functionalities in the networked, automotive control systems. This will be the basis for an interoperable and standardized solution for adaptation and fault handling in AUTOSAR. The SafeAdapt approach also considers functional safety with respect to the ISO 26262 standard.

SafeAdapt provides an integrated approach for engineering such adaptive, complex and safe systems, ranging from tool chain support, reference architectures, modelling of system design and networking, up to early validation and verification. For realistic validation of the adaptation and redundancy concepts, an actual vehicle prototype with different and partly redundant applications is developed.

In this project the author of the thesis worked on:

- Defining the use cases for demonstration the safe adaptation behaviour of the architecture
- Identifying the challenges of applying ISO 26262 to the adaptive systems
- Support to define the safety concept for the new architecture
- Appliance of the contracts proposal to contracts to integrate the software adaptation mechanism within different hardware platforms.

This project has helped to improve the idea of composition within different context. The use of contracts has been applied to assess the verification of the safety requirements. One piece of software implemented as a SEooC is released into two different hardware platforms and contracts has helped assessing safety composition.

8.3.4 OMG Structured Assurance Case Metamodel Standardization Committee

One of the contributions presented on this thesis is the extension of the SACM standard. The contribution has been presented on chapter 4 and the extension has been used while developing the three case studies.

The extension of the metamodel in order to support patterns and modular argumentation along with the GSN mapping has been presented to the SACM standardization committee from the OMG group as an input for future versions.

At the moment of writing this thesis, the author is under discussions for the next version of the standard in order to include some of the proposals presented here.

"Out of clutter find simplicity; from discord find harmony; In the middle of difficulty lies opportunity" – Albert Einstein

9

Conclusions

The present work has introduced a compositional approach to face the challenges arising when dealing with assurance in a component-based environment. The work is based in a model-based approach for assurance specification, a methodology for assurance responsibility decomposition and a contract-based approach to be used while integrating the components.

9.1 Thesis contributions

At the introduction of this thesis when describing the main problems safety-critical systems face when assuring safety standards compliance on component-based systems. In chapter 1 we defined the scope of this thesis on three perspectives:

- Perspective 1: Guidance. It consists of providing guides, formalisation, and methodology. It is a way to structure the knowledge base and expose that knowledge to be used by different stakeholders.
- Perspective 2: Reuse parts/components. This perspective is focused on managing the decomposition of responsibilities of the work at component level and integration at system level.
- Perspective 3: Automation. The aim here is the automation where possible. Provide a methodology and tools to support integration of components into a system from the assurance perspective

In the previous chapters the different challenges for each of the perspectives are presented and potential solutions are proposed. In order to address the compositional assurance problem, we can summarise the solution into these contributions:

- Assurance modelling
- Assurance decomposition
- Contract-based approach for assurance integration
- Tool support for the above

This thesis provides the next technological bridge towards safe and secure safety critical systems. It provides the means to ensure safety as well as the mechanisms required by industry to ensure a safe society while reducing system production costs. The three perspectives ensure an increase both in the safety assurance in industry, while providing

the means for the regulatory bodies to validate safety in industry in an normalized environment, while reducing certification overheads to industry.

9.1.1 Assurance modelling

Chapter 4 has focused on the assurance modelling approach, defining the Common Certification Language metamodels and its use. This is directly linked with the provision of guidance and structuring of knowledge the first perspective of contributions defined on the scope.

Assurance modelling provides us the mechanism to understand the common basis of standards. This is the first step in generating guidelines and formalising compliance with the standards within a company and a project.

The extension of the SACM metamodel is used here in order to specify the assurance cases for the components and the system. They are the central piece of model that serves us to link the different elements. Requirements from the standards are transformed into claims on the argumentation model. These claims are supported by evidence, and the evidences are seen in a form of artefacts that are linked with the evidence model. Those two concepts (claims and evidences) are used by the compliance maps to trace what was requested to what has been done for compliance on a specific project.

9.1.2 Compositional assurance decomposition

The compositional assurance decomposition contribution can addresses as perspectives 1 and 2. It provides us a guideline and the mechanism to decompose the responsibilities associated with a component. We are able to ensure a hierarchy of assurance projects where the responsibilities and tasks can be specified and there is a mechanism to indicate compliance with those tasks.

Assurance decomposition supports the reuse of components as it guides us not just for standards compliance but specifically on the understanding and tailoring of those standards for component assurance.

When dealing with compositional assurance, we have proposed the need to create different assurance projects each of them focused in a component the system has been decomposed into. Also an integration assurance project is required to be responsible for referring to others projects activities and/or artefacts in order to ensure themselves that compliance is not broken. Assurance modelling and the compositional assurance decomposition method support us on this task.

9.1.3 Contract-based approach for assurance integration

This contribution addresses perspectives 2 and 3. On the one hand the contract-based approach supports the integration of reused components and, on the other hand, the proposal supports the identification of assumptions, a very laborious and time consuming task.

Assurance Contracts are defined to ensure incremental compliance once the components are integrated. The objective of this assurance contracts is to ensure the overall

compliance of the system with the selected standards and reference documents such as guidelines or advisory circulars.

The defined approach for assurance contracts specification attempts to balance the need for unambiguity on the composition while maintaining the heterogeneity of the information managed. The claims classification offers an easy method to support the assessment of contract completeness and the structured expressions provide a semi-formal language to specify the assumptions and guarantees of a component

9.1.4 Tool support

In order to fulfil perspective 3 of the scope and support the previous contributions tool support has been developed and applied on the different case studies.

The tool implements the metamodels shown in the assurance modelling chapter and provides editors for each of them. The tool support is not explained on a specific chapter but has been shown in the different chapters and been used on the case studies.

9.2 Relevant Publications

Most of the previous section's contributions have been presented to, and discussed with the scientific community in international workshops, conferences and journals. In this section, we present the articles in which this research has been published.

Table 26 Outline of the contributions and the publications achieved

Contribution	Publication
Assurance modelling	Model-Based Specification of Safety Compliance Needs for Critical Systems: A Holistic Generic Approach Submitted to Information and Software Technology Journal
Assurance modelling	Making Software Safety Assessable and Transparent. EuroSPI^2 2013
Assurance modelling	Safety Case Driven Development for Medical Devices SAFECOMP 2015
Compositional approach	A Preliminary Study towards a Quantitative Approach for Compositional Safety Assurance SSS 2013
Contract-based assurance	Towards a multi-view point safety contract SASSUR 2013
Contract-based assurance	Adequacy of contract grammars for component certification SAFECOMP fast abstract 2013
Compositional approach	Systematic application of ISO 26262 on a SEooC: Support by applying a systematic reuse approach DATE 2015
Tooling	A Tool suite for Assurance Cases and Evidences:

	Avionics experiences EuroAsiaSPI^2
Tooling	An industrial experience in cross domain assurance projects EuroAsiaSPI^2

International Workshop Papers

Alejandra Ruiz, Tim Kelly, Huáscar Espinoza: "Towards a multi-view point safety contract." SASSUR Workshop 2013

International Conference Papers.

Risto Nevalainen, Alejandra Ruiz, Timo Varkoi: "Making Software Safety Assessable and Transparent." EuroSPI^2 2013

Alejandra Ruiz, Huáscar Espinoza, Fulvio Tagliablò, Sandra Torchiaro and Alberto Melzi: "A Preliminary Study towards a Quantitative Approach for Compositional Safety Assurance" 21st Safety-Critical Systems Symposium (SSS), 2013

Alejandra Ruiz Huascar Espinoza, Tim Kelly. "Adequacy of contract grammars for component certification." Safecomp FastAbstract, 2013

Alejandra Ruiz, Alberto Melzi, Tim Kelly: "Systematic application of ISO 26262 on a SEooC: Support by applying a systematic reuse approach." DATE 2015

Alejandra Ruiz, Paulo Barbosa, Yang Medeiros and Huascar Espinoza: "Safety Case Driven Development for Medical Devices." SAFECOMP 2015

Alejandra Ruiz, Xabier Larrucea, Huascar Espinoza: "A Tool suite for Assurance Cases and Evidences: Avionics experiences" EuroAsiaSPI^2 2015

Alejandra Ruiz, Xabier Larrucea, Huascar Espinoza, Franck Aime, Cyril Marchand: "An industrial experience in cross domain assurance projects" EuroAsiaSPI^2 2015

International Journals Indexed in the JCR.

Jose Luis de la Vara; Alejandra Ruiz; Katrina Attwood; Huascar Espinoza; Rajwinder K Panesar-Walawege; Angel Lopez; Idoya del Rio; Tim Kelly: "Model-Based Specification of Safety Compliance Needs for Critical Systems: A Holistic Generic Approach". This publication has been submitted to Information and Software Technology Journal and it is pending for approval.

SASSUR: International Workshop on Next Generation of System Assurance Approaches for Safety-Critical Systems

The SASSUR workshop provides a forum for thematic presentations and in-depth discussions about reuse and composition of safety arguments, safety evidence, and contextual information about system components, in a way that makes assurance and certification more cost-effective, precise, and scalable.

SASSUR aims at bringing together experts, researchers, and practitioners, from diverse communities, such as safety and security engineering, certification processes, model-based technologies, software and hardware design, safety-critical systems, and applications communities (railway, aerospace, automotive, health, industrial manufacturing, etc.).

In this workshop the paper “Towards a multi-view point safety contract”, presents some of the challenges for component-based in relation with safety, and safety assurance, properties and the issues faced by a contract-based approach. Safety is a system property and because of that, it can be hard to define the contribution of components that have an impact on safety. Contract-based approaches addressing safety have been proposed in the past regarding modular safety case development. In this paper we suggested a “multi viewpoint” contract approach where these many aspects are organized to address different stakeholder concerns.

EuroSPI: European System, Software & Service Process Improvement & Innovation

The EuroSPI² conference presents and discusses results from systems, software and services process improvement and innovation (SPI) projects in industry and research, focusing on the gained benefits and the criteria for success.

EuroSPI^2 is a partnership of large Scandinavian research companies and experience networks (SINTEF, DELTA, STTF, FiSMA), the ASQF as a large German quality association, the American Society for Quality and ISCN as the co-coordinating partner. EuroSPI2 collaborates with a large number of SPINs (Software Process Improvement Network) in Europe. In 2015 it becomes European & Asian Systems, Software & Service Process Improvement & Innovation.

According to Computer Science Conference Rank (CORE), this conference is classified as B.

The paper, “Making Software Safety Assessable and Transparent”, highlights the difficulty to software safety assessment in a component. We propose a balanced use of process assessment and product evaluation methods. We should compensate the lack of transparency in software with a more formal development process. Safety cases are an effective approach to demonstrate safety, and then both process and product evidences are necessary

The paper “A Tool suite for Assurance Cases and Evidences: Avionics experiences” describes a specification and an implementation of a flexible tool platform for assurance and certification of safety-critical systems. This tool platform is built upon a comprehensive conceptual assurance and certification framework. This conceptual framework is composed of a common information model called CCL (Common Certification Language) and a compositional assurance approach. The ultimate goal of our platform is to provide an integrated approach for managing assurance cases and evidences resulting from a safety project.

The paper, “An industrial experience in cross domain assurance projects” presents the experience of component reuse in the avionics domain and the use of the tool suite used

on this thesis for the use on a specific case involving a reuse of a software component developed for the railway domain, reuse in the avionics domain.

SSS: Safety-critical Systems Symposium

Safety-critical Systems Symposium is an annual event organized by Safety-Critical Systems Club, an organization that has operated in support of colleagues in the safety community since 1991

The Symposium is for engineers, managers and academics working in the field of system safety, in a variety of roles, and across all industry sectors. It offers wide-ranging coverage of current safety topics, and a blend of academic research and industrial experience, including both recent developments in the field and discussion of open issues that will shape future progress.

The paper, “A Preliminary Study towards a Quantitative Approach for Compositional Safety Assurance” presents preliminary research towards applying a compositional safety assurance approach based on the ISO 26262 concept of SEooC (Safety Element out of Context). In this approach a component must be evaluated against ‘assumed’ operational context conditions in a quantitative manner (based on compatibility/gap analysis), instead of using inspections. Once the component becomes part of a specific system in an actual operational context, the evaluation is optimized by comparing assumed context conditions against actual context conditions. We propose a classification scheme to organize information about assumptions and guarantees and outline a procedure to systematically manage their specification, validation and gap analysis.

SAFECOMP: international conference on computer safety, reliability & security

SAFECOMP was established in 1979 by the European Workshop on Industrial Computer Systems, Technical Committee 7 on Reliability, Safety and Security (EWICS TC7), SAFECOMP has contributed to the progress of the state-of-the-art in dependable application of computers in safety-related and safety-critical systems.

SAFECOMP is an annual event covering the experience and new trends in the areas of safety, security and reliability of critical computer applications. It provides ample opportunity to exchange insights and experience on emerging methods, approaches and practical solutions. It is a one-stream conference without parallel sessions, allowing easy networking.

According to Computer Science Conference Rank (CORE), this conference is classified as B.

The paper, “Adequacy of contract grammars for component certification” moves one step further forward with the creation of a methodology and grammar that incorporates encompasses and helps structure current models of ‘safety contracts’ and propose an assurance contract structure.

The paper, “Safety Case Driven Development for Medical Devices” proposes a methodology to enhance Model-Based System Engineering (MBSE) practice from the safety perspective, encouraging the use of safety cases and providing guidance on how to

show the correspondent traceability to development artifacts. We illustrate our methodology and its usage in the context of an industrial Automated External Defibrillator (AED). The benefits are a clearer mapping of IEC 62304 recommendations and ISO 14971 standardization to the system and safety engineering activities.

DATE: Design, Automation and Test in Europe

DATE is the major international event for design and engineering of Systems-on-Chip, Systems-on-Board and Embedded Systems Software and brings together designers and design automation users, researchers and vendors, as well as specialists in hardware and software design, test and manufacturing of electronic circuits and systems

According to Computer Science Conference Rank (CORE), this conference is classified as B.

The paper , “Systematic Application of ISO 26262 on a SEooC. Support by applying a systematic reuse approach”, presents our experience of the application of the SEooC concept from ISO 26262 to an electric parking system, the automotive case study presented on this thesis. We describe a systematic approach that takes into account the needs for safe reuse of system elements integration into the whole vehicle context

Information and Software Technology Journal

Information and Software Technology is an international technical journal covering software development. It bridges the gap between the theories of software engineering and the application of information technology within organizations. The journal covers the entire area of information processing, from state-of-the-art research, through software development and implementation, to information systems management. The impact factor on 2014 was 1.33

The paper, “Model-Based Specification of Safety Compliance Needs for Critical Systems: A Holistic Generic Approach” paper has been submitted to this journal and its acceptance is still pending. The paper focused on the assurance modelling contribution. It provides a model-based approach for the specification of safety compliance needs for critical systems. The approach is based on a holistic and generic metamodel that 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, which correspond to a model of the structure of a given standard. The resulting models also provide an effective means of structuring and managing safety compliance information.

9.3 Further Work

During the course of writing and undertaking research for this thesis, a number of areas for further research have been identified. This section provides a brief overview of these areas.

The following list includes topics and areas that have been identified for further research in order to widen the scope of the actual work:

- Connection with engineering process and tools
- Applicability to adaptive systems.
- Appliance of Formal Methods
- Run-time assurance

Connection with engineering process and tools

When working on the case studies and specifically on the medical device case study where the development of the system was performed alongside the assurance activities we notice the need for connection the engineering and the assurance ecosystems.

Engineering development is usually done by a team separately from the assurance team; however, the information source is the same, information about the system. In the presented approach, documents, models, designs used to justify the compliance where taken from the system development. There is a need to connect these two worlds on a transparent way.

Applicability on adaptive systems

Adaptive systems are becoming more popular and reliable and could be the new evolution in critical systems when used as fail operational purpose. For example, a failure of an ECU (Electronic Control Unit) in a traditional car can be handled by turning of this unit without losing control of the driving behavior. In contrast, an ECU hosting a break-by-wire application cannot be shut down without losing the ability to break, as long as there is no costly mechanical backup installed. This shift from fail-silent to fail-operational systems poses a great challenge for future automotive systems and to ensure these systems assurance.

Components might end up in a new context due to adaptation. This change t can also be seen as a component reuse, where the possible environments in which they can be adapted are new context of reuse.

A contract-based approach such as the one presented here might be a possible method used to define the possible accepted environments in which a component can end up working. Either if the component ends up being integrated into multiple different systems or if the component is integrated into one system that supports adaptation for use in multiple environments, the components needs to indicate constraints on the working environment in form of assumptions.

Formal Methods

Contracts specification could be improved by a more formal language to express the assumptions and guarantees. This formal language could further reduce the ambiguity in these contracts and enable some formal validation to be performed to ensure satisfaction on the specification.

Applying a formal language in this way could affect the learning curve on the use and became a barrier for adoption. It is necessary that along with this language tool support should be developed.

After developing a formal language for contract specification, a second step would be to define a method for formal validation of the contract. Language formalization and formal validation can be a very complex topic due to the heterogeneous information treated on assurance contract. This diversity is directly connected to the different typology of the analysis that can be applied. Also some of the information treated needs the judgment of domain experts based on their previous backgrounds to provide a valid response.

Run-time assurance

Another important area that can widen the scope of the research is run-time assurance. Fist we should start by identifing those aspects in the systems that are highly dependent with the target environment where the component will end up operating. Once they are identified we are able to delegate some of the verification tasks to an integration phase instead of component design time.

It can also be used to define a monitoring system to ensure that assurance related properties remain in the desired value range during operational time.

Further tool support

During the presentation of the approach presented in this work tooling has also being shown and is a part of this thesis. Tooling has been used on the case studies implementation in order to store the data and for metrics evaluation. However, the tooling has not been the focus on this thesis and during the case studies development, potential new features have been identified in order to improve existing functionality and also to new gaps for laborious tasks have been identified such as transparent assumptions and guarantees import from the component argumentation models into the integration argumentation model,

9.4 Final remarks

Just because a product complies with the standards and follows a well-defined process does not mean that the product is safe.

Standards are made to ensure that best practices of the industry are followed; however, it is still important to ensure deeper safety analysis is undertaken and prepare to the unexpected. Standards will evolve along with the new technologies and improvement in safety analysis.

Standards compliance should not be a barrier for new upgrades, technologies and innovations to be included in safety-critical systems. Certification processes should not impose a delay to put the product in the market or required a big investment from the industry. At the same time certification should ensure the best practices to ensure the safety of a system.

References

[AC 20-148] FAA Advisory Circular: AC 20-148 Reusable Software Components

[AC 20-170] FAA Advisory Circulation AC 20-170 Integrated Modular Avionics Development, Verification, Integration, and Approval Using RTCA/D0-297 and Technical Standard Order-C153

[Antonino et al. 2015] Antonino, Pablo Oliveira; Trapp, Mario; Barbosa, Paulo; Sousa, Luana, "The Parameterized Safety Requirements Templates," Software and Systems Traceability (SST), 2015 IEEE/ACM 8th International Symposium on , vol., no., pp.29,35, 17-17 May 2015

[Arana et al. 2011] N. Arana, H. Espinoza and A. Noguero, eDIANA Project, Deliverable "D6. 3-D Software Engineering Process for Certification Metamodel". 2011.

[ARP 4754A] SAE ARP4754/EUROCAE ED-79, Certification Considerations for Highly Integrated or Complex Aircraft Systems

[AUTOSAR] AUTOSAR project. "AUTOSAR Specification" http://www.autosar.org/. Accessed 19 December 2013

[Benveniste et al. 2012.] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. Henzinger, and K. G. Larsen, "Contracts for System Design", INRIA, Research RR-8147 , Nov. 2012.

[Blanquart et al. 2012] J. P. Blanquart, J. M. Astruc, P. Baufreton, J. L. Boulanger, H. Delseny, J. Gassino, G. Ladier, E. Ledinot, M. Leeman, y J. Machrouh, "Criticality categories across safety standards in different domains", ERTS-2012, Toulouse, pp. 1–3, 2012.

[Bloomfield 2012] Robin Bloomfield, "Assurance Cases. Divide and conquer or divide and fall?", presented at the LAW 2012, 2012.

[Bloomfield et al. 2006] R.E. Bloomfield, S. Guerra, A. Miller et al., International Working Group on Assurance Cases (for Security). IEEE Secur Priv 4:66-68, 2006

[BPMN 2.0] Object Management Group, 'Business Process Model and Notation 2.0'.

[Brown 1998] Alan W. Brown, "From component infrastructure to component-based development", ICSE International Workshop on Component-Based Software Engineering (1998)

[Cancila et al. 2010.] D Cancila, R Passerone, T Vardanega, M Panunzio; "Toward correctness in the specification and handling of non-functional attributes of high-integrity real-time embedded systems"; Industrial Informatics, IEEE Transactions on 6 (2), 181-194

[CESAR D_SP1_R3.3_a_M3] CESAR Project, Deliverable "D_SP1_R3.3_a_M3 Meta-Model Concepts for RTP V"; 2012

[Chevrel 2011] Cedric Chevrel; “Avionics System Certification”, Certification Together conference, 2011

[Clements Northrop 2001] Paul Clements and Linda Northrop; “Software Product Lines: Practices and Patterns”, Addison-Wesley Professional, 2001

[Conmy et al. 2003] P. Conmy, M. Nicholson, and J. McDermid, "Safety assurance contracts for integrated modular avionics", in Proceedings of the 8th Australian workshop on Safety critical systems and software-Volume 33, 2003, pp. 69–78.

[de la Vara 2013] J. L. de la Vara, «Current and necessary insights into SACM: An analysis based on past publications», en 2014 IEEE 7th International Workshop on Requirements Engineering and Law (RELAW), 2014, pp. 10-13.

[Def Stan 00-56] UK Ministry of Defence. Defence Standard 00-56 Issue 4: Safety Management Requirements for Defence Systems, Ministry of Defence; 2007.

[Denney Pai 2012] 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.

[Despotou Kelly 2008] G. Despotou, T. Kelly. “Investigating the Use of Argument Modularity to Optimise Through-life System Safety Assurance” In proceedings of the 3rd IET International Conference on System Safety (ICSS) 2008, 20-22 October 2008, NEC, Birmingham, U.K. Proceedings by the IET.

[Dyson George 1997] Dyson, George, “Darwin among the machines”, Addison-Weslely, Reading, MA, 1997

[DO-178c] RTCA DO-178c/EUROCAE ED-12c, Software Considerations in Airborne System and Equipment Certification

[DO-254] RTCA DO-254/EUROCAE ED-80 Design Assurance Guidance for Airborne Electronic Hardware

[DO-297] RTCA DO-297/EUROCAE ED-124 Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations

[Dodd Habli 2012] 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.

[EAST-ADL V2.1.12] EAST - ADL Association, EAST-ADL Specification Version 2.1.12. 2013.

[Espinoza et al. 2011] H. Espinoza, A. Ruiz, M. Sabetzadeh, and P. Panaroni, "Challenges for an Open and Evolutionary Approach to Safety Assurance and Certification of Safety-Critical Systems", 2011, pp. 1-6.

[Eveleens 2006] R. L. C. Eveleens, “RTO-EN-SCI-176 Integrated Modular Avionics Development Guidance and Certification Considerations,” 2006.

[Falessi et al. 2011] D. Falessi, Shiva Nejati, M. Sabetzadeh, L. Briand, A. Messina. "SafeSlice: A Model Slicing and Design Safety Inspection Tool for SysML". (ESEC/FSE'11), 2011

[Falessi et al. 2012] Falessi, D., et al.: Planning for Safety Evidence Collection. IEEE Software 29(3): 64-70 (2012)

[Fenn et al. 2007-1] J. L. Fenn, R. D. Hawkins, P. J. Williams, T. P. Kelly, M. G. Banner, and Y. Oakshott, "The who, where, how, why and when of modular and incremental certification," in System Safety, 2007 2nd Institution of Engineering and Technology International Conference on, 2007, pp. 135–140.

[Fenn et al. 2007-2] J. Fenn, R. Hawkins, P. Williams, y T. Kelly, "Safety Case Composition Using Contracts -Refinements based on Feedback from an Industrial Case Study", Proceedings of 15th Safety Critical Systems Symposium(SSS'07)

[Flood Habli 2011] M. Flood and I. Habli, 'Multi-view safety cases', in 2011 6th IET International Conference on System Safety, 2011, pp. 1 –6.

[FRESCOR] FRESCOR project "Framework for Real-time Embedded Systems based on COntRACTS" Web: http://www.frescor.org Last Access: 04-05-2015

[Gill 2003] N. S. Gill, "Reusability issues in component-based development", ACM SIGSOFT Software Engineering Notes, vol. 28, no. 4, pp. 4–4, 2003.

[Gorski 2004] J.Gorski, "Trust Case –a case for trustworthiness of it infrastructures", in Proc NATO Advanced Research Workshop on Cyberspace Security and Defence: Research Issues, Gdansk, Poland, 2004

[GSN Standard] Origin Consulting GSN Community Standard Version 1. 2011.

[Haddon-Cave 2009] Charles Haddon-Cave QC. "The Nimrod review. An independent review into the broader issues surrounding the loss of the RAF Nimrod MR2 Aircraft XV230 in Afghanistan in 2006", The Stationery Office, October 2009

[Hawkins et al. 2013] R. Hawkins, I. Habli, T. Kelly, and J. McDermid, "Assurance cases and prescriptive software safety certification: A comparative study," Saf. Sci., vol. 59, pp. 55–71, Nov. 2013.

[Hawkins Kelly 2013] R. Hawkins and T. Kelly, "A Software Safety Argument Pattern Catalogue" University of York, Department of Computer Science Report YCS-2013-482 (2013). Available for download at ftp://ftp.cs.york.ac.uk/reports/2013/YCS/482/YCS-2013-482.pdf

[Hobbs 2011] Chris Hobbs, "Clear SOUP and COTS Software for Medical Device Development", 2011

[Holloway 2013] C. M. Holloway, «Making the implicit explicit: Towards an assurance case for do-178c», ISSC 2013.

[IEC 61508] IEC 61508 - Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems. International Electro technical Commission (2011)"

[IEC 62304] IEC 62304 standard: Medical device software – Software life cycle processes, 2006

[Illarramendi et al. 2014] M. Illarramendi, L. Etxeberria, and X. Elkorobarrutia, "Reuse in Safety Critical Systems: Educational Use Case First Experiences", pp. 417–422, 2014

[ISO 26262] International Organization for Standardization (ISO), ISO26262 Road vehicles – Functional safety, ISO, Nov 2011

[Jackson et al. 2007] D. Jackson, M. Thomas, and L. I. Millett; Book: Software for Dependable Systems: Sufficient Evidence?, Committee on Certifiably Dependable Software Systems, National Research Council, USA.

[Johnson Derek 2011] Chris W. Johnson and Derek A. Robins, «Mith and Barriers to the Introduction of Safety Cases in Space-Based Systems», presented at ". Proceedings of the 29th International Systems Safety Society, Las Vegas, USA. 20, 2011.

[Johnson Robins 2011] C. W. Johnson and D. A. Robins, 'Mith and Barriers to the Introduction of Safety Cases in Space-Based Systems', presented at the ". Proceedings of the 29th International Systems Safety Society, Las Vegas, USA. 20, 2011.

[Kelly 1998] T. P. Kelly, "Arguing Safety - A Systematic Approach to Safety Case Management", University of York, Department of Computer Science, 1998.

[Kelly 2001] T. P. Kelly, "Concepts and principles of compositional safety case construction", Contract Research Report for QinetiQ COMSA/2001/1/1, 2001.

[Kelly 2003] T. P. Kelly, "Managing Complex Safety Cases", in Current Issues in Safety-Critical Systems, F. Redmill and T. Anderson, Eds. Springer London, 2003, pp. 99–115.

[Kelly et al. 2005] Kelly, McDermid and Weaver, "Goal Based Standards Opportunities and Challenges", In Proceedings of the 23rd International System Safety Conference, August 2005, proceedings published by the System Safety Society

[Kelly McDermid 1997] T. P. Kelly and J. A.McDermid, "Safety case construction and reuse using patterns", SAFECOMP'97, 1997,York, UK: Springer

[Klindt 2012] Prof Dr Thomas Klindt, "Technical Standards and their Impact on Product Liability" ISO 26262 Conference, 2012

[Knight 2002] J. C. Knight, "Safety critical systems: challenges and directions", in Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on, 2002, pp. 547–550.

[Kotov 1997] V. Kotov, "Systems of System as Communicating Structures," Hewlett Packard Computer Systems Laboratory Paper HPL-97-124, (1997), pp1-15.

[Kornecki Zalewski 2009] Kornecki, A., Zalewski, J.: Certification of software for real-time safety-critical systems: state of the art. Innovations in Systems and Software Engineering 5(2) 149-161 (2009)

[Larrucea et al. 2013] X. Larrucea, A. Combelles, and J. Favaro, “Safety-Critical Software [Guest editors’ introduction],” IEEE Software., vol. 30, no. 3, pp. 25–27, 2013.

[Ledinot et al. 2012] E. Ledinot, J. M. Astruc, J. P. Blanquart, P. Baufreton, J. L. Boulanger, H. Delseny, J. Gassino, G. Ladier, M. Leeman, and J. Machrouh, «A cross-domain comparison of software development assurance standards», ERTS-2012, Toulouse, pp. 1–3, 2012.

[Leveson 2012] N. G. Leveson, Engineering a Safer World: Systems Thinking Applied to Safety. Cambridge, Mass.: The MIT Press, 2012.

[Machrouh et al. 2012]J. Machrouh, J. P. Blanquart, P. Baufreton, J. L. Boulanger, H. Delseny, J. Gassino, G. Ladier, E. Ledinot, M. Leeman, y J. M. Astruc, «Cross domain comparison of System Assurance», ERTS-2012, Toulouse, pp. 1–3, 2012.

[MACL 2013] Machine-checkable Assurance Case Language http://www.omg.org/cgi-bin/doc?sysa/2012-9-4

[Mars Polar Loss 2000] Jet Propulsion Laboratory, Report on the Loss of the Mars Polar Lander and Deep Space 2 Missions, JPL D-18709 (March 2000).

[MARTE 1.1] Object Management Group, ‘UML Profile for MARTE: Modelling and Analysis of Real-time Embedded Systems 1.1’.

[Matsuno et al. 2010] Y. Matsuno, H. Takamura, and Y. Ishikawa, ‘A Dependability Case Editor with Pattern Library.’, in HASE, 2010, pp. 170–171.

[McCaffery et al. 2014] F. McCaffery, P. Clarke, and M. Lepmets, ‘A Lightweight Assessment Method for Medical Device Software Processes’, in Software Process Improvement and Capability Determination, A. Mitasiunas, T. Rout, R. V. O’Connor, and A. Dorling, Eds. Springer International Publishing, 2014, pp. 144–156.

[McDermid 2001] J. McDermid, "Software safety: where’s the evidence?", Proceedings of the Australian workshop on industrial experience with safety critical systems and software; 2001

[OPENCOSS D1.1]OPENCOSS Project, Deliverable "D1.1 Constraints of the certification process" (March 2012)

[OPENCOSS D2.3] OPENCOSS Project, Deliverable "D2.3 OPENCOSS Platform Architecture.” (2015)

[OPENCOSS D4.4] OPENCOSS Project, Deliverable "D4-4 Common Certification Language: A Conceptual Model" (March 2015)

[OPENCOSS D5.2] OPENCOSS Project, Deliverable "Detailed requirements the OPENCOSS compositional certification approach.” (October 2013)

[OPENCOSS D5.3] OPENCOSS Project, Deliverable "D5.3 Compositional Certification Conceptual Framework." (December 2013)

[OPENCOSS D5.6] OPENCOSS Project, Deliverable "D5.6 Compositional Certification Framework: Methodological Guide" (March 2015)

[OPENCOSS] OPENCOSS Project, URL: http://www.opencoss-project.eu/, Web-Page, Last visit: 2015-03-31

[OTS Guidance 1999] U.S. Department Of Health And Human Services, Food and Drug Administration, Center for Devices and Radiological Health, Office of Device Evaluation, "Guidance on Off-the-Shelf Software Use in Medical Devices." 1999

[Palin Habli 2010] Rober Palin, Ibrahim Habli: Assurance of Automotive Safety - A safety Case Approach; SAFECOMP 2010

[Panesar-Walawege et al. 2010] R. Panesar-Walawege, M. Sabetzadeh, L. Briand, and T. Coq, "Characterizing the chain of evidence for software safety cases: A conceptual model based on the IEC 61508 standard," in ICST'10, 2010

[Papadopoulos McDermid 1999] Y. Papadopoulos and J. A. McDermid, "The potential for a generic approach to certification of safety critical systems in the transportation sector", Reliability Engineering & System Safety, vol. 63, n.º 1, pp. 47-66, jan. 1999.

[polarsys] https://www.polarsys.org

[Quiniou 2011] S. Quinton, S. Graf, y R. Passerone, «Contract-based reasoning for component systems with complex interactions», Verimag Research Report, TR-2010-12, 2010..

[Rasche 2001] T. Rasche, "Development of a safety case methodology for the Minerals Industry – a discussion paper", Minerals Industry Safety and Health Center (2001)

[Redmill 2000] F. Redmill, Safety integrity levels — theory and problems, lessons in system safety. In: 18th safety-critical systems symposium; 2000

[Reuter 2012] Andreas Reuter, "Certification to ISO 26262 –What are the Advantages?", ISO 26262 Conference, 2012

[RTI 2014] Real-Time Innovations, RTI, 'Saving Millions of Dollars in the Development and Certification of Safety-Critical Applications', whitepaper, 2014

[Ruiz et al. 2012] Alejandra Ruiz, Ibrahim Habli, Huáscar Espinoza, "Towards a Case-Based Reasoning Approach for Safety Assurance Reuse. SAFECOMP Workshops 2012: 22-35

[Ruiz et al. 2013_1] 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

[Ruiz et al. 2013_2] A Ruiz, T Kelly, H Espinoza; "Towards a multi-view point safety contract", SAFECOMP 2013-Workshop SASSUR , September 2013

[Ruiz et al. 2015] Alejandra Ruiz, Xabier Larrucea, Huascar Espinoza: “A Tool suite for Assurance Cases and Evidences: Avionics experiences” EuroAsiaSPI2 2015

[Runeson Höst 2009]P. Runeson and M. Höst, ‘Guidelines for conducting and reporting case study research in software engineering’, Empirical Software Engineering, vol. 14, no. 2, pp. 131–164, Apr. 2009.

[Rushby 2002] J. Rushby SRI International CSL Technical Report Modular Certification. June 2002

[Rushby 2007] J. Rushby, ‘Just-in-time certification’, in Engineering Complex Computer Systems, 2007. 12th IEEE International Conference on, 2007, pp. 15–24.

[Rushby 2010-1] J. Rushby, «Formal Methods and Argument-based Safety Cases», in Lectures for Marktoberdorf 2010 Summer School: Software and Systems Safety: Specification and Verification 2010.

[Rushby 2010-2] J. Rushby, «Formalism in safety cases», in Making Systems Safer: Proceedings of the Eighteenth Safety-Critical Systems Symposium, Springer, 2010, pp. 3–17.

[Sabetzadeh et al. 2011] M. Sabetzadeh, D. Falessi, L. Briand, S. Di Alesio, D. Mcgeorge, V. Åhjem, J. Borg. "Combining Goal Models, Expert Elicitation, and Probabilistic Simulation for Qualification of New Technology", HASE'11

[SACM 1.1] Object Management Group, ‘Structured Assurance Case Metamodel (SACM) 1.1’.

[SAFECER D2.2.1] pSAFECER Project, Deliverable: "D2.2.1 & D2.2.2 Specification of the requirements on the generic component model, including certification properties and safety contracts” (2012)

[SAFECER D5.4.1] pSAFECER Project, Deliverable “Definition of cross-domain use case description, use case-specific requirements and assessment criteria” 2012

[Sljivo et al. 2013] I. Sljivo, J. Carlson, B. Gallina, and H. Hansson, «Fostering Reuse within Safety-critical Component-based Systems through Fine-grained Contracts», en proceedings of the International Workshop on Critical Software Component Reusability and Certification across Domains (CSC2013), 2013.

[Sljivo et al. 2014] I. Sljivo, B. Gallina, J. Carlson, H. Hansson, and S. Puri, ‘A Method to Generate Reusable Safety Case Fragments from Compositional Safety Analysis’, in Software Reuse for Dynamic Systems in the Cloud and Beyond, Springer, 2014, pp. 253–268.

[Solingen Berghout 1999] Rino van Solingen and Egon Berghout; “ The Goal/Question/Metric method: a practical guide for quality improvement of software development. Ed. McGraw-Hill. ISBN 007 709553 7, 1999

[SPEEDS D2.5.4] D.2.5.4 Contract Specification Language (CSL); SPEEDS Project; Deliverable; Rev. 1.0.1; April 2008

[Stensrud et al. 2011] Erik Stensrud, Torbjørn Skramstad, Jingyue Li and Jing Xie, (2011): Towards Goal-based Software Safety Certification Based on Prescriptive Standards, International Workshop on Software Certification (WoSoCER 2011)

[Storey 1996] N. Storey, “Safety critical computer systems”, in Addison-Wesley,1996.

[Sutcliffe Carroll 1999] A. G. Sutcliffe and J. M. Carroll, ‘Designing claims for reuse in interactive systems design’, International Journal of Human-Computer Studies, vol. 50, no. 3, pp. 213–241, 1999.

[TECNALIA] Fundación Tecnalia Research & Innovation. www.tecnalia.com

[Vardanega 2009] Tullio Vardanega “Property Preservation and Composition with Guarantees: From ASSERT to CHESS”. ISORC 2009: 125-132

[Viginia]
http://dependability.cs.virginia.edu/research/safetycases/safetycasesexamples.php

[Wagner et al. 2010] S. Wagner, B. Schätz, S. Puchner, and P. Kock, ‘A Case Study on Safety Cases in the Automotive Domain: Modules, Patterns, and Models’, in 2010 IEEE 21st International Symposium on Software Reliability Engineering (ISSRE), 2010, pp. 269 – 278.

[Weaver et al. 2002] R. A. Weaver, J. A. McDermid and T. P. Kelly, “Software Safety Arguments: Towards a systematic Categorisation of Evidence”, Proc.20thInternational System Safety Conference (ISSC), 2002, Denver, USA: System Safety Society.

[Woodman et al. 2001] Mark Woodman , Oddur Benediktsson , Bruno Lefever , Friedrich Stallinger; “Issues of CBD Product Quality and Process Quality” in Proceedings of the 4th International Workshop of Component-Based Software Engineering at 23rd International Conference on Software Engineering, (2001)

[Yani 2011] Anne Yani; “The ‘Extended Airworthiness’ Airbus policy and its impact on certification way of working”. Certification Together Conference, 2011

[Ye Kelly 2004] F. Ye and T. Kelly, ‘Contract-based justification for COTS component within safety-critical applications’, in Proceedings of the 9th Australian workshop on Safety critical systems and software-Volume 47, 2004, pp. 13–22.

[Yin 2013] R. K. Yin, "Case Study Research: Design and Methods", Fifth Edition. Los Angeles: SAGE Publications, Inc, 2013.

[Zeller et al. 2014] M. Zeller, K. Höfig, and M. Rothfelder, ‘Towards a Cross-Domain Software Safety Assurance Process for Embedded Systems’, in Computer Safety, Reliability, and Security, A. Bondavalli, A. Ceccarelli, and F. Ortmeier, Eds. Springer International Publishing, 2014, pp. 396–400.

[Zimmer 2014] Efficiently Deploying Safety-Critical Applications onto Open Integrated Architectures, PhD Theses in Experimental Software Engineering, Bastian Zimmer; 2014

[Zimmer et al. 2014] Bastian Zimmer, Christoph Dropmann, Jochen Ulrich Hanger: A Systematic Approach for Software Interference Analysis. ISSRE 2014: 78-87

ANNEX –
A: Standard analysis tables

All this data is based on the information included on the advisory circular AC 20-148, which presents the software component for reuse as a COTS component and the DO-297when talking about module or application reuse for IMA platforms. The standard ISO 26262 includes information regarding the SEooC, although the guidelines provided are very high level considerations and it is when dealing with the hardware and software qualified component concepts when we can go deeper in the knowledge of the actual requirements for compliance. The IEC 61508 as the standard in which the others are based, also have another relevant concept, the safety manual for the qualified item. All these have provided input to the following tables.

Artefact

Avionics.

Software

Responsible	Artefact	Content
Developer	Plan for Software Aspects of Certification (PSAC) for the RSC	System overview: description of its functions and their allocation to the hardware and software, the architecture, processor(s) used, hardware/software interfaces, and safety features. Software overview: emphasis on the proposed safety and partitioning concepts, for example, resource sharing, redundancy, multiple-version dissimilar software, fault tolerance, and timing and scheduling strategies. Certification considerations: including the means of compliance, the proposed software level(s) and potential software contributions to failure conditions. Software life cycle: The summary explains how the objectives of each software life cycle process will be satisfied. Software life cycle data: the software life cycle data that will be produced and controlled by the software life cycle processes, and the means by which software life cycle data will be made available to the certification authority. Schedule: means to provide the certification

Responsible	Artefact	Content
		authority with visibility of the activities Additional considerations: specific features that may affect the certification process. Define safety concerns: the failure conditions, safety features, protection mechanisms, architecture, limitations, software levels, interface specifications, and intended use of the RSC. Certification liaison process: definition of the process including communication and coordination focal points to all involved stakeholders.
	For each objective	RTCA/DO-178B objective reference; RTCA/DO-178B objective description; Amount of credit being sought (full, partial, or no credit); Assumptions; Means of compliance; and Remaining activities the integrator or applicant must complete.
	RSC software Accomplishment Summary (SAS)	System overview: This section also describes any differences from the system overview in the Plan for Software Aspects of Certification. Software overview: as section before is highlights differences from the software overview proposed in the Plan for Software Aspects of Certification. Certification considerations: restates the considerations mentioned on the PSAC and describes any differences. Software characteristics: states the Executable Object Code size, timing and memory margins, resource limitations, and the means of measuring each characteristic. Software life cycle: explains differences from the software life cycle and software life cycle processes proposed in the PSAC. Software life cycle data: describes the relationship of

Responsible	Artefact	Content
		the data to each other and to other data defining the system. Additional considerations: summarizes certification issues that may warrant the attention of the certification authority and references data items applicable to these issues, such as issue papers or special conditions. Software identification: identifies the software configuration by part number and version. Change history: is a summary of software changes with attention to changes made due to failures affecting safety. Software status: problem reports unresolved at the time of certification, including a statement of functional limitations. Compliance statement: summary of the methods used to demonstrate compliance with criteria specified in the software plans.
	Analysis of the RSC behaviour report	Vulnerabilities, partitioning requirements, , hardware failure effects, requirements for redundancy, data latency, design's constrains
	Agreement from stakeholders for the first application	
	Software Development plan (SDP)	Standards: Identification of the Software Requirements Standards, Software Design Standards and Software Code Standards for the project. Software life cycle: This description is distinct from the summary provided in the Plan for Software Aspects of Certification, in that it provides the detail necessary to ensure proper implementation of the software life cycle processes. Software development environment : software development environment in terms of hardware and software, including: (1) Requirements development method(s) and tools

Responsible	Artefact	Content
		to be used. (2) Design method(s) and tools to be used. (3) Programming language(s), coding tools, compilers, linkage editors and loaders to be used. (4) Hardware platforms for the tools to be used.
	Software Verification Plan (SVP)	It includes the procedures to satisfy the software verification process objectives. This plan should include: Organization: Organizational responsibilities within the software verification process and interfaces with the other software life cycle processes. Independence: A description of the methods for establishing verification independence, when required. Verification methods: the verification methods to be used for each activity of the software verification process. (1) Review methods. (2) Analysis methods (3) Testing methods Verification environment : the equipment for testing, the testing and analysis tools, and the guidelines for applying these tools and hardware test equipment Transition criteria: The criteria for entering the software verification process. Partitioning considerations: methods used to verify the integrity of the partitioning. Compiler assumptions: assumptions about the correctness of the compiler, linkage editor or loader. Reverification guidelines: For software modification, methods for identifying the affected areas of the software and the changed parts of the Executable Object Code. Previously developed software: For previously

Responsible	Artefact	Content
		developed software, if the initial compliance baseline for the verification process does not comply with this document, a description of the methods to satisfy the objectives of this document. Multiple-version dissimilar software : If multiple-version dissimilar software is used, a description of the software verification process activities
	Software Quality Assurance Plan (SQAP)	Environment: A description of the SQA environment, including scope, organizational responsibilities and interfaces, standards, procedures, tools and methods. Authority: A statement of the SQA authority, responsibility, and independence, including the approval authority for software products. Activities: The SQA activities performed for each software life cycle process Transition criteria : The criteria for entering the SQA process. Timing : The timing of the SQA process activities. SQA Records: A definition of the records to be produced by the SQA process. Supplier control: the means of ensuring that sub-tier suppliers' processes and outputs will comply with the SQA Plan.
	Software Configuration Management Plan (SCMP)	Environment: description of the SCM environment used Activities: A description of the SCM process activities that will satisfy the objectives for: Configuration identification, Baselines and traceability, Problem reporting, Change control , Change review, Configuration status accounting, Archive, retrieval, and release, Software load control, Software life cycle environment controls , Software life cycle data controls Transition criteria : The criteria for entering the SCM process. SCM data: software life cycle data produced by the

Responsible	Artefact	Content
		SCM process. Supplier control : The means of applying SCM process requirements to sub-tier suppliers.
	SW Requirements Standards	Methods used for developing software requirements Notations used to express requirements, Constraints on the use of the requirement development tools. Method to provide derived requirements to the system process.
	SW Design Standards	Description method(s) used. Naming conventions used. Conditions imposed on permitted design methods, Constraints on the use of the design tools. Constraints on design, Complexity restrictions
	SW Code Standards	Programming language(s) t used and/or defined subset(s). Source Code presentation standards Naming conventions for components, subprograms, variables, and constants. Conditions and constraints imposed on permitted coding conventions Constraints on the use of the coding tools.
	SQA Records	SQA review or audit reports, meeting minutes, records of authorized process deviations, or software conformity review records.
	Software Verification Results	For each review, analysis and test, indicate each procedure that passed or failed during the activities and the final pass/fail results. Identify the configuration item or software version reviewed, analysed or tested. Include the results of tests, reviews and analyses, including coverage analyses and traceability analyses.

Responsible	Artefact	Content
	Software Requirements Data	Allocation of system requirements to software, with attention to safety-related requirements and potential failure conditions. Functional and operational requirements under each mode of operation. Performance criteria, Timing requirements and constraints. Memory size constraints. Hardware and software interfaces, Failure detection and safety monitoring requirements. Partitioning requirements allocated to software, and the software level(s) of each partition.
	Design Description	
	Source Code	Software identification, including the name and date of revision and/or version Code written in source language(s) Compiler instructions
	Executable Object Code	
	Software Verification Cases and Procedures	
	SCM Records	Configuration identification lists, baseline or software library records, change history reports, archive records, and release records.
	Software Configuration Index	Software product. Executable Object Code. Each Source Code component. Previously developed software included. Software life cycle data. Archive and release media. Instructions for building the Executable Object Code Reference to the Software Life Cycle Environment Configuration Index, if it is packaged separately.

Responsible	Artefact	Content
		Data integrity checks for the Executable Object Code.
	Problem Reports	Identification of the configuration item and/or the software life cycle process activity Identification of the configuration item(s) modified. A problem description A description of the corrective action taken
	Software Life Cycle Environment Configuration Index	Identify the software life cycle environment hardware and its operating system software. Identify the software development tools Identify the test environment used to verify the software product Identify qualified tools and their associated tool qualification data.
	Interface descriptor data	
	Equipment specification	
	List of any RSC subcomponents	
	Instructions for maintenance & calibration	
	Verification Data	List of test cases and procedures affected by any settable parameter
	RSC data sheet	RSC functions Limitations Analysis of potential interface safety concerns Assumptions Configuration Supporting Data Open problem reports Software characteristics Other relevant information that supports the

Responsible	Artefact	Content
		integrator's or applicant's use of the RSC
Integrator	Software Verification Results, verification cases and verification Procedures repeated after integration	
	System level PSAC for the target system	The content includes the same sections as the Component PSAC but this time the information is a system level
	System level SDP	The content includes the same sections as the Component SDP but this time the information is a system level
	System level System level Software Verification Plan (SVP)	The content includes the same sections as the Component SVP but this time the information is a system level
	System level Software Quality Assurance Plan (SQAP)	The content includes the same sections as the Component SQAP but this time the information is a system level
	System level Software Configuration Management Plan (SCMP)	The content includes the same sections as the Component SCMP but this time the information is a system level
	System level Software Configuration Index	The content includes the same sections as the Component SCI but this time the information is a system level
	System level SAS	The content includes the same sections as the Component SAS but this time the information is a system level

Automotive.

SEooC:

Responsible	Artefact	Content
Developer	Item integration and testing plan(s)	Safety activities for product development, testing hardware/software, element integration and item integration Methods and measures during design and integration Interface and interaction between HW and SW Level of robustness Effectiveness of a safety mechanism's diagnosis or failure coverage Performance, accuracy and timing of safety mechanism Consist and correct implementation of interfaces
	Validation plan	Criteria for safety validation Configuration of the item The specification of the validation processes, test cases, driving manoeuvres and acceptance criteria The equipment and the required environmental conditions
	Project plan (refined)	Reference the safety plan
	Safety plan (refined)	Implementation of project-independent safety activities, Definition of the tailored activities Planning: HARA, development activities, supporting processes, verification activities, confirmation reviews and analysis of depended failures
	F.S. assessment plan (refined)	Work products required by the safety plan, the processes required for

Responsible	Artefact	Content
		functional safety and reviewing the appropriateness and effectiveness of the implemented safety measures
	Technical safety requirements specification	External interfaces The constrains The system configuration requirements Ensure consistency with architectural assumptions Dependencies between system, item elements and other systems Response of the systems or elements to stimuli that affect the achievement of safety goals Safety mechanism The transition to the safe state, the fault tolerant time interval
	System verification plan	
	Technical safety concept	
	System design specification	
	HW/SW Interface Specification (HSI)	
	System verification report (refined)	
	Safety analysis report	
	Integration testing specification(s)	
	Integration testing report(s)	
	Validation report	
	Functional safety assessment report	
	Release for production report	

Responsible	Artefact	Content
	Hardware safety requirements specification (including test and qualification criteria)	
	Hardware safety requirements verification report	
	Hardware design specification	
	Hardware safety analysis report	
	Hardware design verification report	
	Specification of requirements related to production, operation, service and decommissioning	
	Analysis of the effectiveness of the architecture of the item to cope with the random hardware failures	
	Review report of evaluation of the effectiveness of the architecture of the item to cope with the random hardware failures	
	Analysis of safety goal violations due to random hardware failures	
	Specification of dedicated measures for hardware	
	Review report of evaluation of safety goal violations due to random hardware failures	
	Hardware integration and testing report	
	Software verification plan	

Responsible	Artefact	Content
	Design and coding guidelines for modelling and programming languages	
	Tool application guidelines	
	Software safety requirements specification	
	Software verification plan (refined)	
	Software verification report	
	Software architectural design specification	
	Software safety requirements specification (refined)	
	Safety analysis report	
	Dependent failures analysis report	
	Software unit design specification	
	Software unit implementation	
	Software verification specification	
	Embedded software	
	Assumed requirements related to design external to SEooC	
	Report of probability of violation of safety goal due to random HW failure	

SW component:

Responsible	Artefact	Content
Developer	Evidence that the SW component complies with its requirements	Show requirement coverage (Software verification Plan, Software verification specification and Software verification report) Cover both normal operating conditions and behaviour in case of failure Results from known errors to show that they do not lead to the violation of safety requirements
	Evidence that the SW component is suitable for its intended	Verification of the validity of the intended use Specification of the software component comply with the requirements of the intended use
	Evidence that the software development process for the component is based on an appropriate standard	
	Software component documentation	Requirements of the software component Configuration description Interfaces description Application manual Software integration description Reaction of the functions under anomalous operating conditions Dependencies with other software components Know anomalies description with correspondent work-around measures
	Software component qualification report	Unique identification of the SW component The unique configuration of the SW component The person or the organization who carried

Responsible	Artefact	Content
		out with the qualification The environment used for qualification The results of the verification measures applied to qualify the software component The maximum target ASIL of any safety requirement that might be violated of the SW component performs incorrectly
	Safety plan refinement	Unique identification of the SW component The maximum target ASIL of any safety requirement that might be violated of the SW component performs incorrectly The activities that shall be carried out to qualify the software component

Hardware component

Responsible	Artefact	Content
Developer	Qualification plan	Identification and version of the hardware component or part Specification of the environment in which the hardware component or part is intended to be used Qualification strategy and rationale Tools and equipment used for implementing the strategy Party responsible for qualification Criteria used to assess the hardware qualification
	Hardware component test plan	Description of the functions Number and sequence of test Requirements for assembly and connections Procedure for accelerated ageing Operating and environmental conditions to be simulated

		Pass/fail criteria established
		Environmental parameters
	Qualification report	The analytical methods and assumptions
		Data from operational experience or testing result
		A rationale for each assumption
		The results of the verification measures applied to qualify the component

Properties

Avionics.

Category	Property	
Installation	Configuration	Interface specification
		Settable parameters specification
		Intended use description
	Environment	Hardware
		Software
Safety	Possible safety problems	
	Safety features	
	Safety related requirements	
	Failure categories	
	Failure conditions	
	Protection mechanism	
	Risk mitigation	
	Assumptions	
Operational	Software levels	
	Operational specifications	
	Operation possible effects (problems)	
	Limitations	
Functional	Architecture & design features	
	Safety architectural function	
Performance	Performance specifications	
	Performance possible effects (problems)	
	Reuse issues	Timing
		Memory Usage
		Resource usage
		Resource items

		Data coupling
		Partitioning
		Protection
		Deactivated code
		Traceability
		Robustness
Reuse -compliance	Analysis of all interfaces	
	Analysis of all settable parameters	
	Analysis of all assumptions of intended use	
	Safety Analysis	
	Applicable credit for reusable test	

Automotive.

Category	Property
Installation	Description on configuration Interfaces Application manual Software component integrations Environment → environmental endurance → tools and equipment →Environmental conditions → limits of these conditions →Environmental parameters Requirements for assembly
Safety	Requirements→Maximum ASIL of any safety requirement Known anomalies → work around measures Requirements coverage Functions → functional performance Failure modes → failure mode distributions → failure models Diagnostic capability Limits of use
Operational	Operational effects (problems) Operational conditions → limitations Behaviour Dependencies
Functional	Requirements for testing equipment Pass/fail criteria for test Analytical method use Rationale for each assumption
Performance	Procedure for accelerated ageing

	Robustness Conditions/ test cases
Reuse -compliance	Criteria to assess qualification

Industry

Category	Property
Installation	Input/output interfaces → interface constrains Hardware /Software configurations → hardware run-time environment → compilation – link system Maintenance requirements Installation instructions Compatibility with other systems Configurable elements → methods for configuration
Safety	Safe state Design Instructions or constrains observable to prevent systematic failures Anomalies Information for external diagnosis of failure mode Failure mode → failure rates→ failure mode of the diagnostic internal → diagnostic test interval → outputs initiated by the diagnosis
Operational	Evidence for systematic capability to provide functionality Constrains on the use → Assumptions on the use
Functional	Functions specification
Performance	Behaviour analysis Periodic proof test Hardware fault tolerance
Reuse/compliance	Competence → minimum degree of knowledge expected of the integration (example tools) Degree of reliance

Process

Avionics.

Responsible	Process/Activity
Developer	Plan development
	Verification procedures
	Certification liaison process

	Inform any deviation from plans
	Make reviews and adjustments
	Prescribe activities to gain full credit for the installation
	Identify verification activities that integrator must repeat
	Analysis of any potential functional, operational, performance and safety effects
	Analysis of all interfaces
	Analysis of all settable parameters
	Define Installation or integration procedures
	Define periodic maintenance and/or calibration
	Process for making some data available to the applicant, without supplying the data to the applicant
	Identify and maintain data to support changes to the RSC
	Retain and maintain a list of all integrators and applicants buying or using their components
Integrator	Activities prescribe by developer to gain full credit for the installation
	Repeat verification activities define by developer
	Retest where new setting or parameters may affect the requirements, code, function, performance, or protection features
	Analysis of data coupling and control coupling of the RSC
	Development of new test cases and procedures to complete all test and test coverage objectives
	Open problem reports on the RSC and analysis of any potential functional, operational, performance and safety effects
	Integrate plans into own software lifecycle
	Produce a system level plans
	Evaluate safety, operational, performance and functional impacts
	Follow the approved plans and standards
	Validate the assumptions made by RSC's developer
	Validate and verify the throughput, timing, memory usage, resource usage, and other resource items

	Report in-service problems with the RSC
	Investigate the in-service experience related to the RSC
	Establish a legal agreement with the RSC developer
	Submit all SCIs, SAS and other required software lifecycle data to the certification authority

Automotive.

Responsible	Process/Activity
Developer	Planning the activities that shall be carried out to qualify the software component
	Specify the software/hardware component or SEooC
	Provide evidence that software component complies with its requirements
	Show requirement coverage I part 6 clause 9
	Verify both normal operating conditions and behaviour in the case of failure
	Verify results that no known error lead to violation of safety requirements
Integrator	Measure structural coverage to evaluate completeness of test cases
	If necessary specify additional test cases or provide rational to ensure completeness to test cases
	Verification of qualification of a software component