Collaborative Large-scale Integrating Project

Open Platform for EvolutioNary Certification Of Safety-critical Systems

Detailed requirements the OPENCOSS compositional certification approach D5.2

Work Package:	WP5: Compositional Certification
Dissemination level:	Public
Status:	Final
Date:	October 31st , 2012
Responsible partner:	Bart Jacobs (inspearit, formerly DNV)
Contact information:	Bart.jacobs@inspearit.com

Contributors

Names	Organisation
Georgio Tagliafelli	RINA Services SpA
Joost Gabriels	Eindhoven University of Technology
Alessandra Martelli	Intecs
Jose Luis de Vara	Simula Research Laboratory
Daniela Cancila	ATEGO France
Bart Jacobs	Inspearit (before DNV ITGS)
Bernhard Sputh	ALTREONIC
Philippa Conmy, Oleg Lisagor, Katrina Attwood	University of York

Document History

Version	Date	Remarks
V0.1	2012-07-09	First ToC
V0.2	2012-07-18	ToC update
V0.3	2012-07-25	First contributions Section 1
V0.4	2012-09-25	Contributions to Sections 1 - 3
V0.5	2012-10-04	Update V0.4
V0.6	2012-10-17	Major revision Sections 1 - 3
V0.7	2012-10-29	Changed structure, revised requirements
V0.8	2012-10-31	Intermediate version
V1.0	2012-11-xx	Final version

TABLE OF CONTENTS

List of Figures

List of Tables

Executive Summary

This document D5.2 Detailed requirements for the OPENCOSS compositional certification approach is the second deliverable of task T5.1. It continues the work from D5.1 Baseline for the compositional certification approach of OPENCOSS and presents detailed requirements for a compositional certification approach.

To this end, D5.2 presents three main sections that provide input for the compositional certification framework in the next WP5 task.

The introduction defines the scope in more detail then – but including recommendations from – the baseline in D5.1. WP5 will research the reuse of certification data for components being moved between systems and domains. Therefore it aims to develop techniques to promote and exploit harmonisation and reuse of logical reasoning which is used to provide assurance of the safety of components, (sub)systems and of the gathered evidence. The section includes motivations and drivers for compositional certification, as well as a set of pragmatic and research challenges and constraints.

The discussions and considerations section starts with considerations from related, mainly technical, work packages. It continues an introduction to compositional construction of arguments, with an introduction to modular GSN as a solution for argumentation notation. An approach for the proposed solution is explained in more detail in three areas of compositional argumentation techniques: component arguments, safety argument module development and argument strategies.

The detailed requirements are derived from the first two sections. Bearing in mind the scope and the discussions and considerations section, the requirements are structured into four areas for further research:

1) requirements for argument architecture, argumentation types and mechanisms for expressing claims
2) requirements for details and development of safety argument modules
3) requirements for managing and analysing emergent properties within the argument
4) and finally, requirements that relate to more general project requirements, for implementation and relationships to other work packages.

1 Introduction

The OPENCOSS project aims to facilitate reuse of certification and analytical data for safety critical products. This document motivates and develops requirements for this aim, specifically for an approach to composing independently developed certification data. Within this document the term "evidence" is used to denote the output of an analysis process (for example some software test results), whereas the broader term "certification data" is used to include the evidence, plus information on its provenance and trustworthiness. We assume that the latter is presented as an "argument" about the evidence, which can also be reused within different projects. The term "assurance assets" is synonymous with "certification data".

With this in mind, at the heart of the OPENCOSS project is the need to structure and store knowledge concerning safety assurance and certification in such a way that stakeholders from across the project's target domains can share, use and reuse this knowledge in a clearly-managed, harmonised way. At present, this harmonisation is difficult to achieve, since the standards and practices deployed within each of the target domains (and, indeed, the practices deployed within individual companies or organisations) differ markedly from one another. The ways in which safety is analysed, demonstrated, certified/accepted, documented, managed, conceptualised and discussed in the various domains differ as a result of the lengthy histories of separate evolution of safety praxis within each domain, which has largely been conducted in isolation from other domains. This isolation between the domains is a principal challenge for the OPENCOSS project, which aims to reduce the costs inherent in recurrent safety certification when components or subsystems are reused across projects, organisations or domains by facilitating the informed reuse of assurance assets, such as evidence and arguments [1]. As outlined in [1], OPENCOSS envisages several different scenarios for the reuse of assurance arguments and the evidence artefacts that support them:

i. Reuse of a safety case or evidence artefact for a component or subsystem within a new system, possibly across domains;
ii. Reuse of (part of) a safety case in order to demonstrate compliance to another standard, within the same domain;
iii. Reuse and/or improvement of a safety case in order to regain confidence in the safety of a system, for example after a catastrophic accident;
iv. Reuse of a safety case for a complete system in a new domain
v. Reuse of an evidence artefact for the next version of the safety-critical system for which it was originally developed, within a project or organisation.

WP5 is concentrating on the scenarios represented in Figure 1, re-using certification data for components being moved between systems and domains.

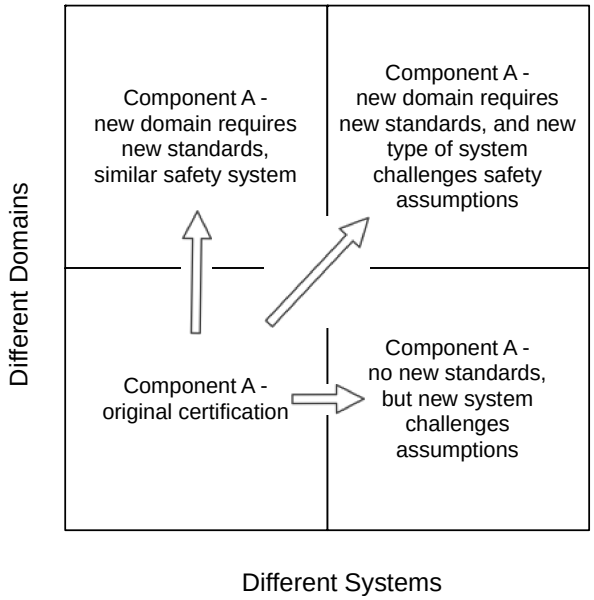

Figure 1: Effect of Component Reuse in Different Systems and Domains

Several motivations for this harmonisation and reuse can be identified:

a) Cost reduction. Safety analysis and the preparation of safety arguments is one of the most expensive aspects of the development of any safety-critical system. However, it is important to note that, in some domains, certification data for system components (which could in principle be reused without a negative impact on safety) must be regathered due to technical and administrative constraints. For example, domain standards may legally require that reused components are re-certified and re-analysed for use in a new system. Where these constraints do not apply, however, it may be possible to argue that re-analysis of a component is not required, since this would simply mean re-verifying existing analysis results. In such cases, reuse of analysis data and reasoning about safety is likely to lead to reductions in the cost of development.

b) Reduced time-to-market for systems. Safety analysis and system justification are extremely time-consuming activities. Similarly, the process of securing regulatory approval for a system is extremely lengthy. By reducing the time needed for these activities to a demonstration that the analysis results and logical reasoning are suitable for reuse in the new context, it should be possible to curtail these timescales.

c) (Possibly) improved system safety. Well-managed reuse of components means that system developers should have access to more information about the reused components (including in-service performance and failure data), and a greater understanding of their tolerances and limitations. In principle, this should mean that developers are able to produce safer systems using the reused components.

d) Market trends strongly suggest that many future embedded systems will be comprised of heterogeneous, dynamic coalitions of systems of 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 systems

Work Package 5 of OPENCOSS is concerned with the development of techniques to promote and exploit this harmonisation and reuse of the logical reasoning which is used to provide assurance of the safety of components, subsystems and systems and of the evidence (e.g. analysis results, tests, performance and

failure data etc.) which is used to support this reasoning, in a safety case context. In practice, evidence is largely the focus of Work Package 6, which aims (among other things) to provide means for configuring and characterising evidence in such a way as to support the claims made in safety arguments. Work Package 5 is mainly confined in scope to the argumentation side of composing certification data. There will be a need to be considerable interplay and harmonisation of the work produced in Work Packages 5 and 6, to ensure that the arguments defined can be supported by the evidence.

The approach adopted in Work Package 5, is one of compositional certification in a safety-argument framework, using the Goal Structuring Notation (GSN) [4]. The basic approach, its motivation and challenges are described in some detail in Section 2 below, but the following brief account will serve to motivate the discussion in the remainder of this section. The compositional certification technique is based on the development of a high-level safety argument to justify claims made about the safety of a safety-critical system. This system-level argument is formed from fragments (or 'modules') of safety arguments, which capture the context, safety justification and essential characteristics of the evidence called on to support the safety claims made at the system level. The logical structure of the argument is carried in the *composition* of these safety argument fragments, which are combined to form a coherent logical argument to justify the complete system. Contracts are used to link the safety argument fragments and to ensure compatibility between the fragments, by assessing the fragments' contribution to the overall argument in terms of rely-guarantee relationships. The logical balance of the whole argument structure is thus maintained.
For systems at a given granularity of detail (e.g. software level), the claims which will need to be addressed, the nature of the evidence put forward and the reasoning which it supports are likely to be relatively generic: standards in the OPENCOSS target domains essentially require demonstration of the same kinds of safety properties.[1] This implies that the high-level system safety argument can be more-or-less generic (although reconfigurable for a given standard or domain, as appropriate).

Ideally, safety argument modules should be associated directly with a tangible component of the system, and should be confined to discussion of that component, its characteristics, behaviour, and any assumptions made about other components. This would simplify reuse, since the safety justification (the argument) could simply "travel with" the component as part of a reusable "certification pack", in the same way that analysis evidence can (to some extent) do. Note that, when we use the term 'component', we refer to a self-contained part, combination of parts, subassemblies or units, which contribute to one or more functions of a system. The term "component" is defined intentionally broadly and may cover parts of the system that range from basic items to major sub-systems. Also, the definition of this term is intentionally broad and covers parts of the system that range from major sub-systems to the smallest engineered items. To avoid confusion the term "component" refers to parts of the engineered system or its design whereas the term "safety case module" refers to parts of the system safety case. Unless explicitly stated otherwise, the term "module" is typically used in this document as shorthand for "safety argument module" - a well-defined reusable part of the safety case. Each component may have one or more modules

[1] Note that, although the domain standards differ in terms of the actual techniques they mandate or recommend for the gathering of evidence to support the safety justification, the nature of the claims does not differ. It is (part of) the concern of WP 4 to provide a means by which techniques can be 'mapped' in terms of the reasons why they are carried out (i.e. what they are trying to demonstrate), and it is (part of) the concern of WP 6 to capture information about the results of the analysis (the evidence) in terms of these objectives.

associated with it, but other modules may not be directly related to components but be about other system properties.

Unfortunately, though, the structure of a safety argument does not necessarily equate to the physical or logical design of the system. The safety argument is a logical, rhetorical entity, calculated to demonstrate (i.e. to persuade a reader of) the safety of the complete system, rather than simply being a description of the system. As such, the argument needs to discuss cross-cutting concerns – such as overall resource balancing -, or to justify the safety of high-level functions which might be spread across several logical (architectural) or physical components: in short, the safety of a system does not equate simply to the sum of the safety of its constituent parts. It is important to note then, that in describing safety argument modules or fragments, or the contracts between them, we do not necessarily reflect the exact structure of the system itself, as it is designed or built.

The authors have established that there are significant business drivers for the compositional certification of safety-critical systems in aviation/avionics, automotive and railway domains via survey, inspection of other internal work, such as the OPENCOSS D2.1 report [1], and various discussions with the project consortium partners as well as within our wider networks of collaborators. Whilst different industrial organisations may express those drivers and motivations differently, they tend to centre on a fundamentally similar set of issues:

- Minimising, wherever possible, the costs of certification (whilst at least maintaining the rigour of the overall process and safety of the systems) through enabling systems integrators to take 'certification credit' for previously assessed / certified components.
- Reducing the re-certification effort and costs for components reused across the boundaries of industrial domains and jurisdictions of different standards, regulators or assessors (when components have been previously certified in another domain/jurisdiction).
- Controlling the cost of re-certification following system change and, in particular, ensuring that re-certification efforts are, at worst, proportionate to the extent of change (and its impact) rather than to the size of the whole system.
- Different approaches for certification between the different domains for certification of safety-critical systems.

In discussions with project partners, we have established that industrial practices with respect to compositional certification vary to great extent not only between different industrial domains, but also between different organisations within each domain. However, across all domains, and with only rare exceptions (such as IAWG consortium in the UK), certification artefacts are captured by means of mainly textual reports with reconciliation of artefacts typically performed in fully manual manner, and adequacy of composition assured through informal manual inspections. This usually results in lengthy, costly, and labour intensive processes. The time and cost overheads associated with the compositional certification are further exacerbated by the lack of support tools. Currently, companies tend to adapt general document management tools to support management of certification artefacts; these tools are often bespoke and inconsistent across the supply chain.

1.1 High-Level Challenges for Compositional Certification within OPENCOSS

In order to support the basic requirements for a compositional certification technique, there are a number of 'pragmatic challenges', involving issues which must be addressed in order to facilitate the adoption of compositional certification methodologies by industry and also involving the integration of compositional certification within the overall approach to certification and reuse proposed by the OPENCOSS consortium – and 'research challenges', which are issues to be addressed at the research level to ensure that reliable composition of safety arguments and evidence artefacts can be achieved.

1.1.1 Pragmatic Challenges

1. Support from, or compatibility with, existing standards or certification authorities. It should be noted that, at present, compositional certification is an "evolving" approach: although regulatory guidance and best practice in some of the OPENCOSS target domains (notably automotive and avionics) has made some headway towards addressing compositional approaches to system development and assurance, the authorities' attitudes to these techniques is still essentially untested. In developing a compositional approach, OPENCOSS must ensure that it seeks the advice and 'buy-in' of relevant agencies, for what is essentially an industry-led approach.
2. Establishment of a common basis for safety across the domains. At present, approaches to the demonstration of safety across the target domains differ markedly, both in terms of the techniques employed and the nature of the processes and arguments put forward. OPENCOSS seeks to harmonise these approaches, by pointing out the similarities in the underlying models of intent and justification inherent in them. The bulk of this harmonisation work will be done in the development of a common framework for certification (including the Common Certification Language (CCL) in WP 4. The generic system-level argument to be produced as the basis for the compositional argument in WP 5, however, must reflect and embody this underlying model. For this reason, close collaboration is required within the project to ensure that WP 5 can 'buy in' to the common framework produced in WP 4, and also to ensure that relevant industrial partners can agree with the harmonised approach presented in the generic (though tailorable) argument.
3. Defining an appropriate granularity of "system" and "components". In order for the compositional certification approach to be successful, it must be possible to define a generic argument of safety to form the basis of the compositional approach. The granularity of the high-level safety claims is crucial here – they need to be expressed in sufficient detail as to be generalizable to all 'target systems', and to reflect the highest-level claims of such systems. While it is relatively easy to see that such claims are generalizable for relatively low-level systems, such as software systems, it may be that there is insufficient commonality between systems at a higher-level of abstraction to make the postulation of general claims feasible. The scope of the case studies to be used as demonstrators for the OPENCOSS approach needs to reflect this concern.
4. Ensuring the compositional certification technique and toolset developed in WP 5 is compatible with the overall approach adopted in OPENCOSS and with the OPENCOSS Platform.

1.1.2 Research Challenges

1. The ability to compose evidence in order to assess its overall contribution to the argument. This challenge will need to be addressed by close collaboration between WPs 5 and 6.
2. Ensuring that the functionality provided by a component (together with the confidence the available evidence gives us in this functionality) will meet a high-level safety claim made in the argument. System safety does not equate to the sum of the safety of its constituent parts. There is a subtle distinction that must be considered here: compositional design requires the matching of component functionality to design requirements. Compositional certification is about the matching of *data*, for the purposes of compliance and to ensure a safe design.
3. Clarifying the relationship between argument claims and system requirements. It is rarely, if ever, the case that claims made in safety arguments relate entirely and exclusively to safety requirements defined for the system.
4. Making the reasoning structures genuinely reusable. Is it desirable to “reuse” a complete safety argument, or is it important to realize that we must fashion it anew each time, in order to ensure logical consistency?

1.2 WP5 Constraints

This section lists a set of assumptions and constraints for the research in WP5, based on a need to manage expectations and make the problem tractable, whilst building a solid foundation upon which a fully worked industrial solution can be based.

1.2.1 Automation

Full automation and formalisation is unlikely to be feasible (or desirable) for compositional certification. This is because arguments are inexact, and safety is not an absolute property which can be quantified. Therefore, human review and judgement is ultimately required to assess whether a system is "acceptably safe" and this stems from an understanding of the argument and compatibility of evidence. However, OPENCOSS WP5 should attempt to provide some structured and formalized process support to the tasks of safety case module matching and composition. This way, the OPENCOSS infrastructure may enable substantial reductions in recurring safety certification costs. It is expected that use of controlled language and terminology (i.e. CCL developed by WP4) within the statements of argument structures could be beneficial for parsing of the structures and would provide the basis for development of some assistance tools. The authors are not aware of any pre-existing research in this area.

1.2.2 Cross-domain support

As noted previously, there is a drive to reuse computer components across systems in multiple domains. At present, re-certification costs mean that most benefits of reuse are counteracted by the need to re-analyse components - often producing no new information about that component. However, sometimes a new context actually leads to an unsafe condition, simply due to different system level behavioural requirements. Part of the challenge for WP5 is gathering enough contextual information about a component in order to understand and reveal any incorrect assumptions which might lead to unsafe behaviour. The large number of potential domains, and their differing safety properties and associated

hazards, mean that this is a huge area. Pragmatically, WP5 will limit itself to examining cross-domain issues for partners represented within the consortium, i.e. automotive, aerospace and railway.

1.2.3 Compatibility with the CCL

WP5 needs to provide the necessary methods and supporting tools for the management compositional certification used in the safety certification of critical systems and also to pay particular attention to the interdependency with the CCL. This means ensuring the terms/concepts used are compatible with the CCL as well as ensuring the cross domain/standards translation theory works for safety argumentation.

1.2.4 Use of software contract principles to define safety argument module interfaces

In discussions with project partners, we have established that industrial practices with respect to compositional certification vary to great extent not only between different industrial domains, but also between different organisations within each domain. However, as noted, across all domains, and with only rare exceptions (such as IAWG consortium in the UK, see [7]), certification artefacts are captured by means of mainly textual reports with reconciliation of artefacts typically performed in fully manual manner, which results in lengthy, costly, and labour intensive processes. We have therefore proposed an adaptation of software contract ideology (with defined rely/guarantee conditions - see [7]) to manage arguments and assurance for a component. The safety argument modules will have a number of claims which require support from other argument modules, and also that offer claims to other modules. Therefore the nature of these contracts will differ greatly from software contracts. For example, a software contract typically contains formally specified conditions, e.g. a pre-condition of x > y which shows that all input values of x are assumed to be greater than a value y [5]. Run time checks, and static analysis, can be performed automatically to ensure that these conditions are not breached.
However, a claim within a safety argument is specified in natural language, e.g. "pressure valve releases when pressure over 5 milibars". This claim may be required to a module with a specific safety requirement to mitigate against a pressure build up, e.g. "explosion prevented due to controlled release of pressure". A safety argument contract will be needed to reconcile these two claims, and will be in the form of another argument. The following table compares the two methods.

Software contract feature	GSN Assurance contract equivalent
Pre-condition	Away goal (requires support from another module)
Post-condition	Public goal (can be used by another module)
Run-time checking via automated tools and exception handling	Bespoke safety argument contracts, which reconcile natural language claims in public and away goals and ensure contexts are compatible.

Table 1: Comparison of Software Contracts and Safety Argument Contracts

The mission in of WP5 in relation to the OPENCOSS platform is:

- Understand how to capture each component's assurance contract and how to propagate the contracts for certification acceptance by other components
- Identify the necessary contractual information

- Consider emergent properties or unexpected interactions which may arise during safety case integration.

2 Discussion and considerations

The purpose of this section is to scope the problem further, by considering current solutions, tools and other sources of information both within OPENCOSS and also external. Our proposed enabling technology for compositional certification is to use the GSN with modular extensions. GSN provides a means to present arguments and justifications. These arguments must be supported by evidence (such as test data etc.). As noted previously, a safety case is constructed of both arguments and evidence. A main aim is to use a safety case as a means to support compositional certification. WP5 is largely concerned with linking and assessing disparate information using compelling arguments, but in order to do so it must consider evidence - hence it is very closely linked to, and must be compatible with, WP6 which looks at chains of evidence.

Another consideration is how to support a safety engineer using these approaches within the OPENCOSS platform. Developments of argumentation strategies is largely process based, and uses natural language, therefore it cannot be automatically generated. However, some argument patterns, and expert advice software which guides the user through the process, can be provided via the platform. Our aim is to ensure that the more theoretical and research oriented output from WP5 can be input into the tool.

The first part presents considerations from OPENCOSS Requirements and Architecture design in WP2, and considerations from the other technical work packages are presented. The second part provides an introduction into modular GSN, followed by three areas of compositional argumentation techniques.

2.1 Considerations from related OPENCOSS work packages

This section looks at the relationships between WP5 and other WPs within OPENCOSS, examining how they relate and shape the research and requirements for research.

- WP2 - this WP is developing the high level requirements and software architecture for the OPENCOSS platform. Several use cases have been developed which are considered in WP5.
- WP4 - this WP is developing a common certification framework, and a means of mapping certification concepts between domains, in order to help with cross-domain certification and reuse.
- WP6 - this WP is examining evidence, links between evidence types and how they evolve.
- WP7 - this WP is concerned with Transparent Certification and Compliance-Aware Development Process.

2.1.1 Considerations from high-level requirements and architecture design (WP2)

WP 2 [2] is the main source for the detailed (research) requirements. It contains high-level requirements for OPENCOSS. In order to select the right high-level requirements for WP5, the draft architecture [3] provided guidance. The selected requirements that are related to compositional certification are elaborated in more detail in section 3.

2.1.2 Considerations from Compositional Certification Language (WP4)

WP4 is building a series of certification models, and developing a CCL, with the intention of translating concepts and methods through a series of domains. For example, to support a developer in understanding

how certification data for a component developed for a railway system (using relevant railway standards), would meet (or fall short of meeting) certification requirements for standards in the avionics domain.

The meta-models being developed include concepts such as Objectives, Safety Processes, and Standards and are broadly grouped as shown in Figure 2. Models are instantiated for particular domains (based on the meta-models) and mappings between them are created as part of a model driven engineering approach. Running through all these models and meta-models is the CCL, which looks at translation of concepts (which may not map exactly or simply due to different terminology and practice in the various domains). In addition, the CCL provides means of expressing safety claims in the various domains.

The argumentation processes and patterns developed in WP5 will need to be complaint with the CCL. In addition, we need to examine ways to map claims in a safety argument from one domain to another, at a minimum to try and highlight areas where the safety argument is challenged (and hence understand our confidence in meeting new certification requirements and also whether we are able to justify the safety of the system in the new domain).

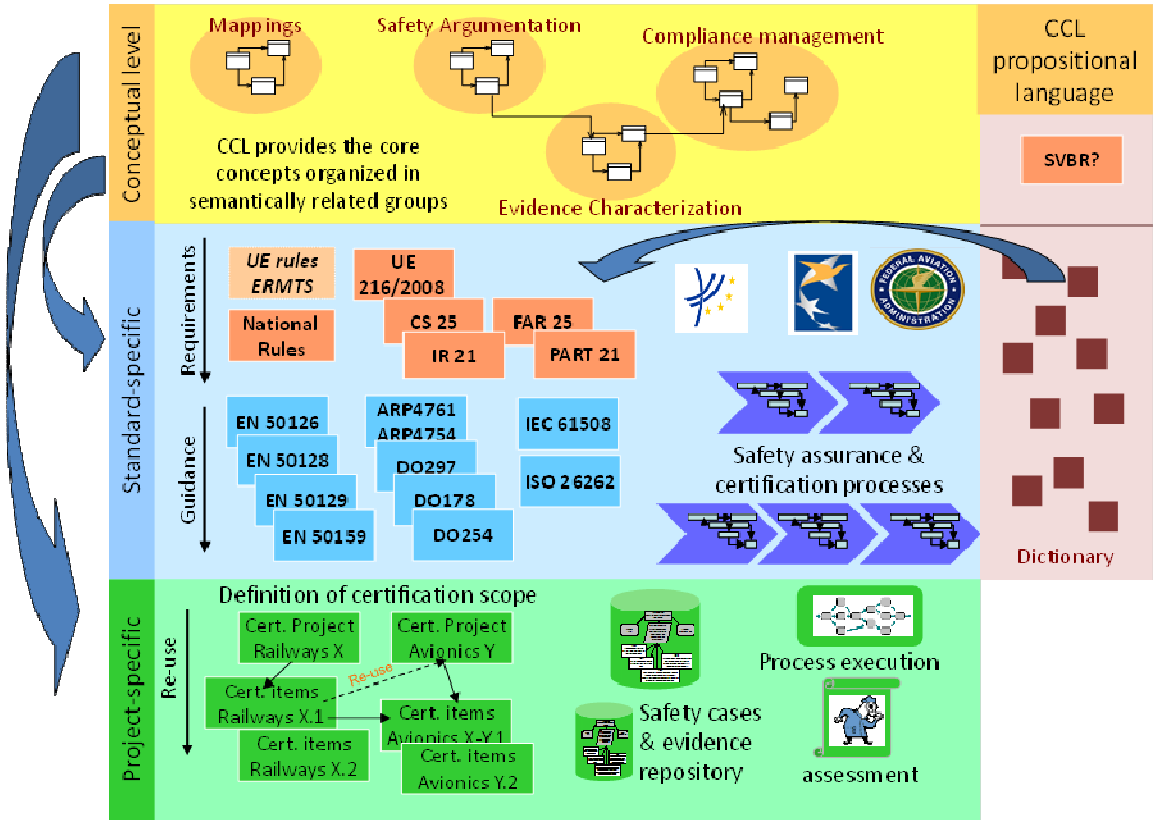

Figure 2: WP4 Common Certification Model and Concepts

2.1.3 Considerations from Evolutionary Evidential Chain (WP6)

OPENCOSS WP6 (Evolutionary Evidential Chain) aims to design and develop tool support for management of evidence-related information in safety assurance/certification projects. This information includes both evidence items of projects (e.g., artefacts resulting from the development of a safety-critical system such as hazard analyses, requirements specifications, and V&V results) and other safety-related elements and information that might have to be created and provided to show system safety (e.g., safety cases and safety arguments).

WP6 will mainly deal with the management and evolution of chains of evidence. Such chains correspond to pieces of evidence that are related, and whose relationship must be adequately kept in order to demonstrate system safety. For example, information corresponding to hazards, safety requirements, design, source code, and testing can belong to a chain of evidence. As a part of the management of chains of evidence, WP6 will focus on the analysis and the needs of situations in which evidence evolves and that can make a chain of evidence become inadequate. That is, a change of a piece of evidence can affect the adequacy of other pieces of a chain of evidence, and thus affect the adequacy of the chain. For example, if source code changes, then testing results might become inadequate.

A chain of evidence can be inadequate not only because of the change of a piece of evidence, but also because of the lack of pieces in the chain. This scenario occurs, for instance, during the development of a safety-critical system. Artefacts are created in a given order, thus all the necessary pieces of a chain of evidence might not be available at a given moment. For example, and in order to fulfil the requirements/objectives of a safety standard, V&V results can be necessary so that a chain of evidence is adequate, but such results would be available after source code. As a result, it might be known that V&V results are necessary so that a chain of evidence that includes source code as a piece of evidence is adequate, and the chain will not be so until the V&V results have been included and linked.

Changes in a chain of evidence might affect other safety-related elements beyond its pieces of evidence. If a piece of evidence changes, then the arguments and claims supported by the piece might be affected. Consequently, it would be necessary to analyse if, for instance, the confidence in the arguments and in the fulfilment of the claims have been negatively affected. If so, new evidence might have to be provided to gain and provide confidence in system safety.

In relation to compositional certification, a scenario in which a chain of evidence can evolve is component reuse. Safety evidence for a component can be included in the chains of evidence and the safety case of a system that reuses the component. Other safety-related elements of a component such as arguments might also be reused. When reusing a component, it might be necessary to provide new evidence in order to have adequate chains of evidence. For example, new evidence might have to be provided to show safe operation of the component in the operation conditions of the system.

Issues resulting from component reuse that WP6 needs WP5 to study are:

- Evidence-related information needs to enable and facilitate component reuse
- Needs in argumentation and thus in safety case development
- Other possible needs in safety documentation

In summary, the compositional -based approach for certification defined in WP5 should try to address all these needs in order to integrate WP5 and WP6 results. The solution(s) defined in WP5 will constrain WP6 work.

From a conceptual perspective, and as a result of the work in the Compositional Conceptual Certification Framework (task T5.2), it might be necessary to include certain information in the safety cases/documentation of components in order to adequately specify component contracts and safety case modules. As a consequence, the evidence model of a safety assurance/certification project for a component would have to include this information. Otherwise, it would not be possible to determine if

(safe) composition of the component is possible or adequate, according to the conceptual framework defined in WP5.

From a technological perspective, when using the OPENCOSS platform and aiming to reuse safety information of a component (e.g., evidence and arguments), WP6 infrastructure will provide WP5 infrastructure with this information, and then WP5 infrastructure will have to determine if (1) more information should be provided, and/or (2) information should be structured in a given, in order to gain confidence in and show system safety. Lack of information might not imply that component reuse is inadequate, but that enough confidence in safe reuse might not exist. In summary, WP5 will define concepts and means to gain confidence in safe component composition/reuse, and they should be supported by WP6 infrastructure.

Aspects that are related to both evidence (and chains of evidence) evolution and compositional certification, and that thus WP5 and WP6 might have to jointly study, are:

- Implications of component reuse in the chains of evidence of a system
- How evidence evolution affects component-based safety elements
- How evidence evolution is constrained by component-based safety elements
- What characteristics/properties of evidence influence compositional certification
- How compositional certification is related to different types of evidence (D6.1 contains an initial evidence taxonomy)
- How compositional certification is constrained by different types of evidence
- The relationship between chains of evidence and argumentation from a component-based view
- The possibility of automatically deriving or structuring contract information/safety case modules for a component from the available, stored information of the component in WP6 infrastructure

The possibility, need, and priority of addressing these aspects will have to be further discussed in the upcoming WP5 and WP6 tasks.

2.1.4 Considerations from Transparent Certification and Compliance-aware Process (WP7)

WP7 aims at defining a safety certification management infrastructure to support the certification process. This process will be interwoven with the development and safety assurance processes by allowing developers to assess where they are with respect to their duties to conform to safety practices and standards, and still to motivate them to see the effective progress of the work and level of compliance.

WP7 will deal with a transparent certification to expose and make explicit as much as possible the overall certification process to all involved actors. This will help those actors to take immediate actions to increase safety, and reduce the costs of the certification process. It will also deal with a compliance-aware development process for the OPENCOSS platform, to rely on external tools to define and execute the process. Such tools typically use BPMN, SPEM or similar process languages to define project processes. The OPENCOSS platform by the mean of WP7 will implement an API that will allow such tool to report to the platform the correct execution of the process. The platform will then be able to compute the conformance of such execution regarding the involved standards.

The approaches to reuse certification data for components being moved between systems and domains, can contribute to expose and make explicit a transparent certification process. The chosen argumentation processes and patterns developed in WP5 should fit with the choices that are made in WP7 for compliance-aware processes. A challenge for WP5 with regards to WP7 is to take into account the compliance-aware processes when re-using certification data for components being moved between domains.

2.2 Introduction to Modular GSN

OPENCOSS (largely in WP5) is looking at compositional construction of arguments, using the GSN. This notation has been chosen as it has explicit extensions which allow arguments to be produced in modules per component - providing justifications and guarantees (backed with evidence) on an individual basis. The notion of modular safety cases has been introduced to facilitate effective development and maintenance of a safety case. Inspired by the notions of modularity in software and system engineering disciplines, modular safety cases are constructed from independently developed modules, with well-defined interfaces. The principal elements of the notation are shown in Figure 3 in the form of example instances of each concept from a safety argument:

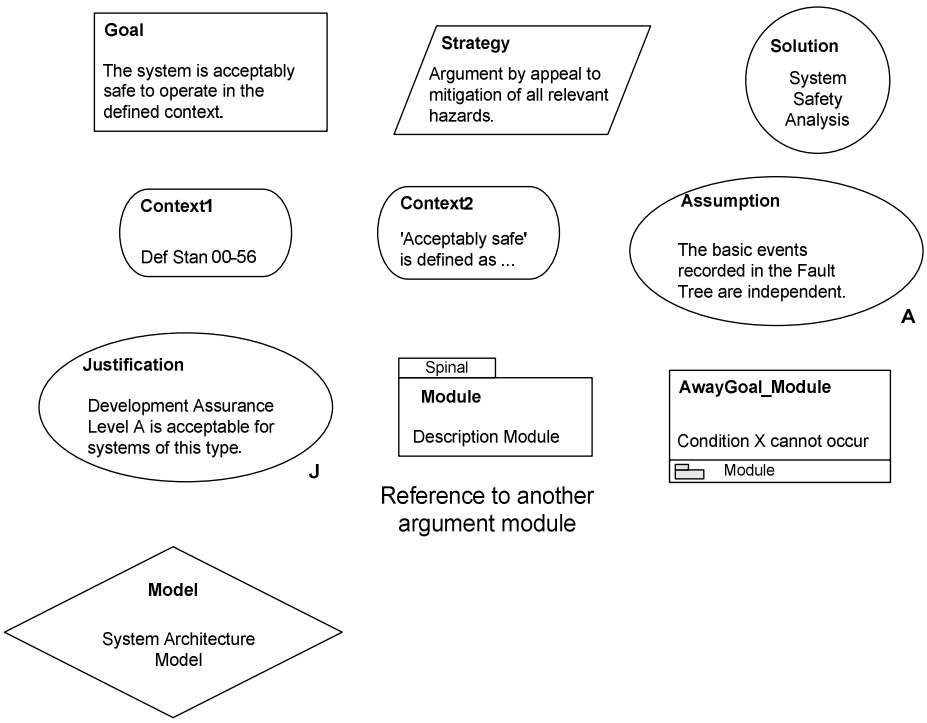

Figure 3: GSN Symbology

When these elements of GSN are connected together, they are said to form a **goal structure**. Sections of Goals (containing argument claims we wish to make) can be collected in modules (as shown in the symbol labelled "spinal" (see also [6]). The modules can be reused within a specific system safety argument, along with bespoke arguments about that system as necessary. For example, a component may have a specific Worst Case Execution Time, which is demonstrated via statistical testing. Arguments about this WCET may include information about the platform used for testing, confidence in the results (how many tests were run) and information on the testing technique used itself (how reliable it has been in the past). This information is reusable in multiple systems. In a specific system context, several WCETs may need to be

reconciled as part of an overall system schedule, in order to meet a specific safety requirement. The latter bespoke argument may be via what is known as a Safety Case Contract which connects various argument modules together [8].

An example of a broad high level structure to a modular argument is shown in the figure below. This argument has reused modules about an operating system and an application running on that OS, with claims reconciled via a safety argument contract. In addition, these are linked to a bespoke system level argument which also needs to link to claims in the modules.

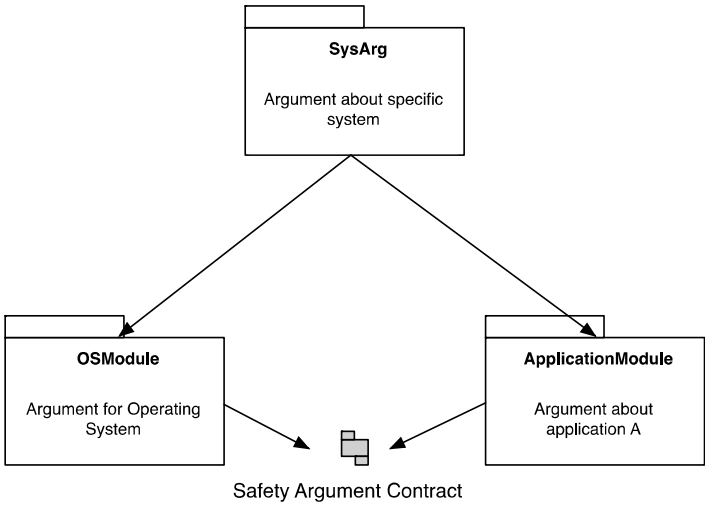

Figure 4: High Level Modular Argument Structure and Architecture

If safety case modules are developed fully independently, with no coordination at the onset, it is highly unlikely that an adequate safety case for an integrated platform can be compiled in a 'bottom-up' fashion. This is due to a number of factors which make matching public and away goal matching difficult as follows:

- Claims are expressed as natural language, and as such may contain ambiguities in the way they are written.
- Claims may be expressed at a different level of abstraction about a component, due to the style of the argument developer. For example, software claims may be made at a low-level of detail (e.g. about freeness of boundary violations for an individual function) or at a higher level of abstraction (e.g. about general use of coding sub-sets and practices which mean that none of the software would have such violations, but it isn't explicitly stated).
- All argument claims are made based on a number of assumptions and within a given context. This is captured in the argument (via the special symbols shown above) but these need to be compared for each module to ensure that claims which may superficially appear to match do in reality.

A more effective process model would require a generic safety case architecture to be established first, to identify key responsibilities of individual modules. In addition, establishing broad reusable principles for style and claim expression would mean it is easier to link claims together. However, since components may be reused across multiple projects, or in different types of systems this approach is not feasible in all cases.

One of the pragmatic challenges faced by the compositional safety case development is initial "start-up" costs associated with establishing a safety case module independently, and a methodology that allows better production of modules, such that they can be put together more effectively at a later stage. It is

possible that for some classes of systems and/or for some features of system development processes and supply chains, these initial efforts and costs will outweigh the benefits of compositional certification. The OPENCOSS project should seek to obtain better understanding of the key parameters of these trade-off decisions and, if possible, to provide guidance for the users of the OPENCOSS platform. Whilst both the notion of safety cases and the application of structured argumentation techniques (such as Goal Structuring Notation) to safety case representation and development are well established and have reached industrial maturity, the concepts of modular and compositional safety case development are still an area of active research.

2.2.1 Potential Approach

There are three areas of compositional argumentation techniques within which broad principles should be established to assist in their reuse and also composition.

1. Bespoke, specific arguments associated with a component about its properties and associated evidence. Whilst the detail of the argument will differ from component to component (an argument for a tyre would be substantially different to that for a piece of software), guidance on how to write claims in an unambiguous way can be provided.
2. Argument strategies/structures or patterns of logic which give consistent and more thorough guidance as to:
    a. Producing component arguments which cover all likely relevant details (e.g. failure rates, static analysis etc.).
    b. Producing the component argument in such a way that it can be more easily combined with other argument modules and built into a system argument
3. Developing system argument strategies (logical breakdown of concepts) and architectures which link the component arguments.

These are now considered in more detail, to lead into the requirements developed in section 3 of this document.

2.2.2 Bespoke Component Arguments

As previously discussed in 2.2, claims within a safety argument are developed in natural language. Constraining the way these are expressed will help in the comparison between them.

2.2.3 Safety argument module development

It is assumed that, loosely, each component (as a reusable unit of code/ programmable hardware or combination of both) will have been examined and analysed against a specification and in the context of a set of assumptions (e.g. a software library with a set of formal requirements, written to be used in an automobile engine). Associated with the component will be one or more safety argument modules, containing fragments of argument, either linked to existing evidence (a "guarantee" linked to by a public goal), or pointing out where gaps in evidence exist or important assumptions have been made (a "rely" expressed as an away goal). These may be a set of assumptions about a particular system and/or a more general idea of how it may be used in the future. Arguments include the evidence quality/clarity and our confidence in it, and that a component really does what it is specified to do. It is important to note that not

all evidence may be reused, and that part of the purpose of the reusable arguments is to flag potential issues to developer looking to use the component in a new context. One challenge with component arguments modules is identifying which claims to represent as public goals, i.e. which should be offered for other argument modules. If we offer too many goals, then the interface becomes difficult to manage and there will be too many safety case contracts required. However, subtle details may be missed if the public goals are at too high a level within the argument, and too broad in their scope.

An additional challenge is to look at broad similar types of component (e.g. software, programmable logic, interlocks), and see if patterns for information that should be in the argument module can be provided. This will again help with the composition of arguments, ensuring matching of relevant claims about similar properties e.g. failure rates.

A final challenge is to ensure that assumptions and contextual information about the component, and the environment in which evidence was gathered and analysis performed, is expressed clearly and unambiguously.

2.2.4 Argument Strategies

There are multiple strategies and patterns for approaching the development of system safety arguments [9]). For example, an argument may be written such that each system hazard is examined in turn, looking at how the risk is appropriately mitigated. This is very compelling to review system safety, however as components within the system may address multiple hazards, it can be difficult to separate evidence about them. An alternative is to look at each component in turn and its contribution to system safety. With this approach hazardous contributions by components may be less explicit. As this is effectively the approach we need to have for OPENCOSS (with modular arguments per component to be reconciled) we need to ensure that the safety requirements are made as visible as possible.

In addition to this we note that safety cases can contain three basic themes within their arguments:

- 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

All of these themes need to be considered for WP5 arguments.

2.2.5 Additional considerations

Much of this section of the document has considered composition of argument modules, partly for reuse and partly with the general problem of composition from scratch. As noted in section 1, part of the reuse problem is to move components to another domain. In this situation some of the concepts referred to in claims may be known by different names, or different approaches to gathering evidence needed. WP4 is producing methods for translation of concepts between domains and standards. The argumentation needs to be written in a way that allows translation within all elements, particularly claims. Further, we need to be able to identify where translation is not possible and where there are missing or mismatched assertions from one domain to another. This issue is represented in the next figure.

There are other issues to consider too, for example the quality of the evidence and the visibility we have of it. A commercial off the shelf (COTS) component may be used in a safety critical system, but the manufacturer only release evidence about it with additional charges. We need to express the argument in a way that makes clear that our confidence in the evidence may be influenced by factors other than just technical ability of the safety assessors.

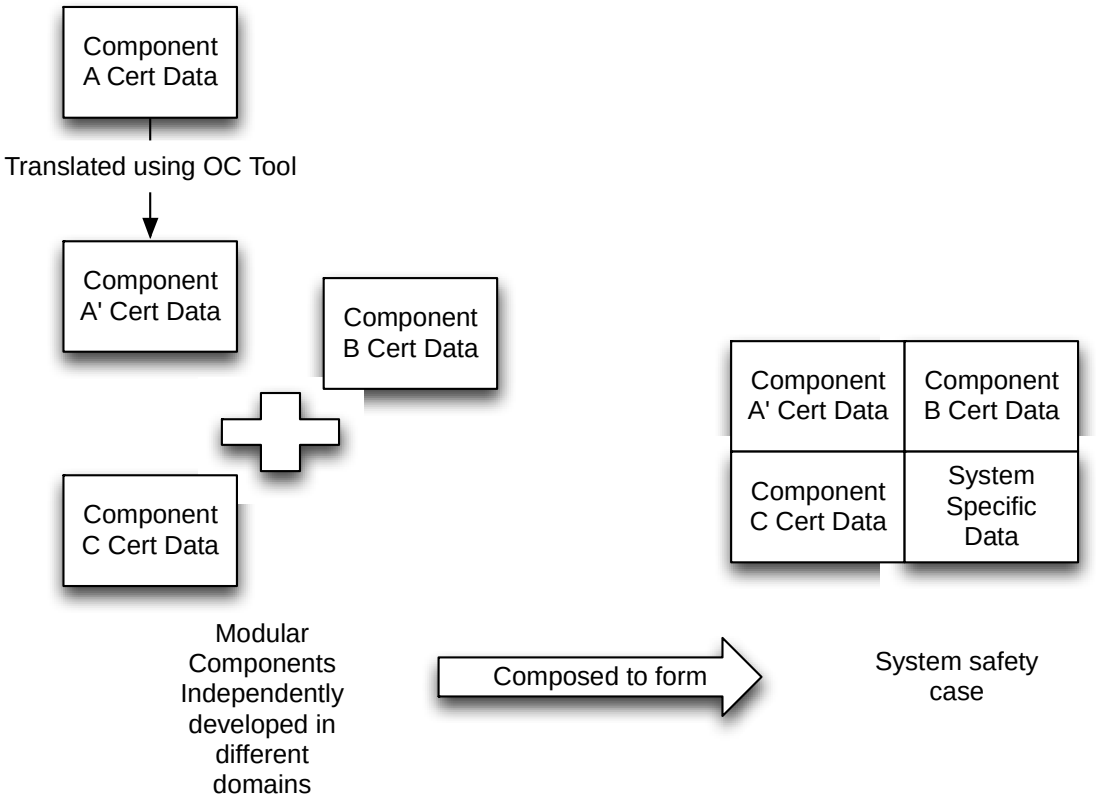

Figure 5: Modularity with cross domain reuse of certification data (OC stands for OPENCOSS)

3 Requirements for compositional certification

Based on the earlier introductory text (and D5.1), we have expressed the need to provide assurance arguments in a modular (probably per component) basis, relating to the reuse of evidence across or within projects and domains. The GSN argument notation has been proposed as a solution, as it is – to the authors' knowledge – the only argumentation notation which provides explicit support for the construction of modular arguments. Bearing in mind the OPENCOSS requirements, we have identified the following areas which require further research:

- High-level compositional argument methods - considering the overall argument architecture, the types and strands of argument required, and mechanisms for expressing claims;
- Low-level detail on how to develop safety argument modules - particularly how to capture rely/guarantee (or contract) based argument structures which can be used to develop the high-level arguments;
- Managing and analysing emergent properties within the argument - looking at unintended interactions, conflicts and compromises as well as looking for hidden problems.

These are considered in turn below with a set of research requirements. A strand running through these requirements is the need for risk, confidence and compliance to be addressed. A further orthogonal concern is the types of change likely to be encountered – changes to a particular component, or moving the component to a new domain or system. The theoretical basis behind this research will be tested via appropriate case study, and will lead to the development of tool support (such as via an expert or advisory system, as described in 1.2.1).

Each requirement for the compositional argumentation approach to be developed in WP5 is represented as follows (the proforma is taken from OPENCOSS deliverable D4.2, which provides requirements for the Common Certification Language):

Title - expressed in exact terms that can be shown to be either met, partially met, or met not at all.

XX-XX – ID Feature XX, requirement XX	
Description	A more detailed statement of the requirement. It must be used in further stages for instance for requirements validation, and it usually should contain the word 'shall', 'should', etc.
Rationale	Complementary information providing further explanation or justification of the requirements.
Stakeholders	Roles within the OPENCOSS framework which are supported (e.g. Safety assessor, Assurance manager, Safety manager [2])
Status	Current status of the requirement; Proposed, Cancelled, Met, Partially met, Not met at all.
Priority	Priority of the requirement, in terms of MoSCoW (Must have, Should have, Could have or Won't have)

3.1 High level compositional argumentation requirements

These requirements relate to the "building blocks" of an argument, without getting into too much technical detail. They also relate to the notion of overall processes of argument construction.

3.1.1 The OPENCOSS Approach shall provide a consistent and constrained means for the expression of safety argument claims

01-01	
Description	Safety-argument claims should be expressed in a consistent and constrained way, and using the language/terminology developed in the CCL. Consistency means that levels of abstraction (in terms of design detail), artefacts, and methods in different claims can be compared in a meaningful way. This will necessarily lead to some constraints on expression.
Rationale	In order for argument modules to be composed successfully, it is essential that the language in claims in the different modules matches, so that the overlaps and gaps between the modules can be properly identified. Argument modules are developed in the context of the requirements/expectations/culture of a particular standard or group of domain-specific standards. If the modules are to be successfully composed, it is important to understand the overall logical structure within which the claims, argument and evidence are put forward. Arguments are targeted towards the demonstration of the satisfaction of safety requirements. It is therefore important to be able to ensure consistency between the expression of requirements as claims between modules, to ensure that modules for composition can be seen to cover compatible requirements.
Stakeholders	Safety engineer, Safety assessor
Status	Proposed
Priority	M/S/C/W ("Assurance concepts" will need to be defined as the project work progresses. In the first instance, the expectation is that the CCL will model concepts likely to appear in safety requirements, generic and particular hazards, evidence artefacts and 'argument concepts')

3.1.2 The OPENCOSS Approach shall provide a consistent and constrained means for the expression of safety argument contracts

01-02	
Description	Matching claims between safety argument modules, or rather ensuring that inexact matches do not affect the overall argument logic or (more importantly) the safety of the system, is done via an additional argument, known as a "contract". This requirement relates to the need to express contracts using consistent language structures, expressed using the terminology provided by the CCL. The concepts used in contracts are likely to be the same as those used in claims, but there may need to be some additional concepts relating to the argument structures themselves (i.e. rely-guarantee relationships).
Rationale	Consistent use of language to express contracts between modules will help ensure the appropriateness and assess the limitations/viability of the composition.
Stakeholders	Safety engineer, Safety assessor

Status	Proposed
Priority	M

3.1.3 The OPENCOSS Approach shall provide a consistent and constrained means for the expression of contextual information used in safety arguments

01-03	
Description	Safety-arguments refer to information concerning the context within which the argument is assumed to be valid, e.g. the operating environment of the system, domain or standards used. These can be expressed using noun-phrases defined in the CCL, to provide a means for a consistent expression of language, but there will need to be consistent syntactic structures in addition. Context here refers to assumptions made by the claim in its original argument use. Again, it is likely that the concepts necessary will be identical to the ones in the claims - e.g. hazards.
Rationale	Managing the context in which a claim or module is stated, and ensuring that it can be appropriately reused is in many ways the key challenge for compositional certification. Consistent means of expression is essential for a clear understanding of the context to support comparison between the modules/candidate modules and assess overall safety.
Stakeholders	Safety engineer, Safety assessor
Status	Proposed
Priority	M

3.1.4 The OPENCOSS Approach shall provide a consistent and constrained means for the expression of assumptions used in safety arguments

01-04	
Description	Safety-arguments often rely on assumptions about the behaviour of entities within the context within which the argument is assumed to be valid, e.g. the operating environment of the system, the domain or standards used. These assumptions can be expressed using noun-phrases expressed in the CCL, to provide a means for a consistent expression of language, but there will need to be consistent syntactic structures in addition. Assumptions provide part of the logical basis on which the claims to which they apply are stated and are valid.
Rationale	Managing the logical basis on which a claim or module is stated, and ensuring that it can be appropriately reused is in many ways the key challenge for compositional certification. Consistent means of expression is essential for a clear understanding of the context to support comparison between the modules/candidate modules and assess overall safety.
Stakeholders	Safety engineer, Safety assessor
Status	Proposed
Priority	M

3.1.5 The OPENCOSS Approach shall develop a library of reusable pattern-based structural templates for modular arguments

01-05	
Description	Safety arguments often make use of standard forms of logical reasoning – claim-decomposition, contextual information and evidence – to substantiate or address particular claims. It is possible to abstract these standard forms into reusable argumentation patterns, where claims and evidence can be instantiated for a particular product in a particular context. These patterns might be domain- or standard-specific. The user is then able to combine patterns to create a safety argument (or a substantial art thereof). Associated with these templates will be methods and processes for their instantiation and development.
Rationale	Using a planned architecture for a safety argument (e.g. based on the design of a traditional three layer stack for software) will help reduce the risk of incompatibility between modules and their claims.
Stakeholders	Assurance manager, Safety manager
Status	Proposed
Priority	M

3.1.6 The OPENCOSS Approach shall provide a means for managing change within a modularised argument

01-06	
Description	If a component is changed within a system, thus leading to a change in that component's argument module, the impact of this change on the overall system safety argument should be minimised. If the component is to be moved to another domain, or to another project governed by another standard, or is to be deployed in a different regulatory context, then there is a need to assess how the argument module is impacted, using the model/translations developed in WP4 as part of the CCL, and how the impact of the change propagates to the system argument in the new argument context. WP5 will need to develop a means of impact analysis for argumentation to assess the ramifications of the change on the overall logical coherence of the (system-level) safety argument.
Rationale	The reuse of components across projects and domains is a primary business driver for the OPENCOSS project, but represents a considerable risk to the assurance of the system into which the component is reused. There will need to be a means to manage or indicate the impact of change within the argument.
Stakeholders	Safety manager, Safety assessor
Status	Proposed
Priority	M

3.2 Lower-level safety argument module requirements

These requirements relate specifically to the development of safety argument modules, and their interfaces.

3.2.1 The OPENCOSS Approach shall provide a consistent and constrained means for the expression, and identification, of public goals

02-01	
Description	The claims which should presented on the modular argument interface as "public" need to be identified, and expressed in a constrained way, using the language/terminology defined in the CCL.
Rationale	In order for argument modules to be composed successfully, it is essential that the language in claims in the different modules matches, so that the overlaps and gaps between the modules can be properly identified.
Stakeholders	Safety assessor, Assurance manager
Status	Proposed
Priority	M

3.2.2 The OPENCOSS Approach shall provide a consistent and constrained means for the expression, and identification, of 'away' goals

02-02	
Description	The claims made in 'away' goals should be expressed in a constrained way, using the language/terminology defined in the CCL.
Rationale	In order for argument modules to be composed successfully, it is essential that the language in claims in the different modules matches, so that the overlaps and gaps between the modules can be properly identified.
Stakeholders	Safety assessor, Assurance manager
Status	Proposed
Priority	M

3.2.3 The OPENCOSS Approach shall provide a consistent and constrained means for arguing about evidence characteristics

02-03	
Description	WP6 will provide models for evidence characterisation. The argument modules will be compatible with these models, and will address and the important elements within them, using the terminology defined in the CCL.
Rationale	It is essential that the approach be compatible with the work done in WP6.
Stakeholders	Safety Manager
Status	Proposed
Priority	M

3.2.4 The OPENCOSS Approach shall provide a means for managing the impact of change within an argument module

02-04	
Description	If a component is altered, then the impact of that change within the components argument module should be minimised. If the component is moved to a new domain then the changes required to the argument module also need to be assessed, using the model/translations developed in WP4 as part of the CCL.
Rationale	The component argument module needs to be robust to inevitable changes to the module, minimising rework and retaining the logical coherence of the argument module in context.
Stakeholders	Safety engineer, Safety manager, Safety assessor
Status	Proposed
Priority	M

3.3 Requirements to manage emergent properties or unexpected interactions which may arise during the integration of argument modules into a system-level safety argument

Whilst it may be possible to perform a pairwise comparison of argument modules, and various claims, there is a need to assess whether the sum of these comparisons still provides a compelling and useful argument overall.

3.3.1 The OPENCOSS Approach shall provide a means for the overall assessment of evidence supporting the argument

03-01	
Description	To provide a means for assessing our overall trust in the composed evidence used to support the argument structure, possibly with some methods for flagging and identifying contested claims, in an implementable way.
Rationale	The degree to which the compositional safety argument is convincing relies on the trust we have in the composed evidence which supported the claims and inferences it contains. There is a need to address the adequacy of the composition of evidence here. Where there are areas where this trust is compromised, it is essential that some indication of the impact of the problem is provided.
Stakeholders	Safety engineer, Assurance manager
Status	Proposed
Priority	M

3.3.2 The OPENCOSS Approach shall provide a means for the overall assessment of risk represented by the composed argument

03-02	
Description	To provide a means for assessing our overall trust in the system risk described by the composed argument structure, possibly with some methods for flagging and identifying contested goals, in a way that is implementable.
Rationale	We need to be sure that the composition of claims in the overall structure addresses all

	relevant system-level risks. Where there are areas where the logical completeness of the overall risk argument is compromised, it is essential that some indication of the impact of the problem be provided.
Stakeholders	Safety assessor
Status	Proposed
Priority	M

3.3.3 The OPENCOSS Approach shall provide a means for the overall assessment of confidence within the composed argument

03-03	
Description	To provide a means for assessing our overall confidence in the composed argument structure, possibly with some methods for flagging and identifying contested goals, in a way that is implementable
Rationale	Argument confidence is a subjective entity, and is not something that can be entirely calculated automatically (although some automated calculation of confidence may be possible).
Stakeholders	Assurance manager, Safety assessor
Status	Proposed
Priority	M

3.4 General Requirements

These requirements relate to more general project requirements, for implementation and relationships to other work packages.

3.4.1 The methods and processes developed should be implementable within the OPENCOSS platform

04-01	
Description	The OPENCOSS platform will include support for assisting the composition of assurance arguments. The processes and methods developed in WP5 provide input to this.
Rationale	The processes and methods developed in WP5 should be both a) expressed as unambiguously as possible b) expressed in terms that are compatible with those employed in the OPENCOSS platform.
Stakeholders	OPENCOSS platform
Status	Proposed
Priority	M

3.4.2 The methods and processes developed should be compatible with approaches developed in the other OPENCOSS work packages

04-02

Description	WP5 has strong links with: • WP4 for the CCL • WP6 to describe evidence • WP7 looking at processes
Rationale	There is an obvious requirement for consistency across the project.
Stakeholders	OPENCOSS platform
Status	Proposed
Priority	M

3.4.3 The methods and processes developed should be validated using appropriate case study material

04-03	
Description	Industrial partners have provided different sets of case study material. This will be used within WP5 to both develop and validate the methods, mechanisms and processes.
Rationale	The requirements for WP5 are primarily about developing techniques and processes as part of a research process. These must be validated, both as proof of concept, and also (ultimately - beyond the life of the project) in an industrial context.
Stakeholders	OPENCOSS platform
Status	Proposed
Priority	M

3.5 Alternative view of requirements

The following table represents these requirements against four alternative viewpoints coming from WP2 draft architecture view [3] and resulting use cases.

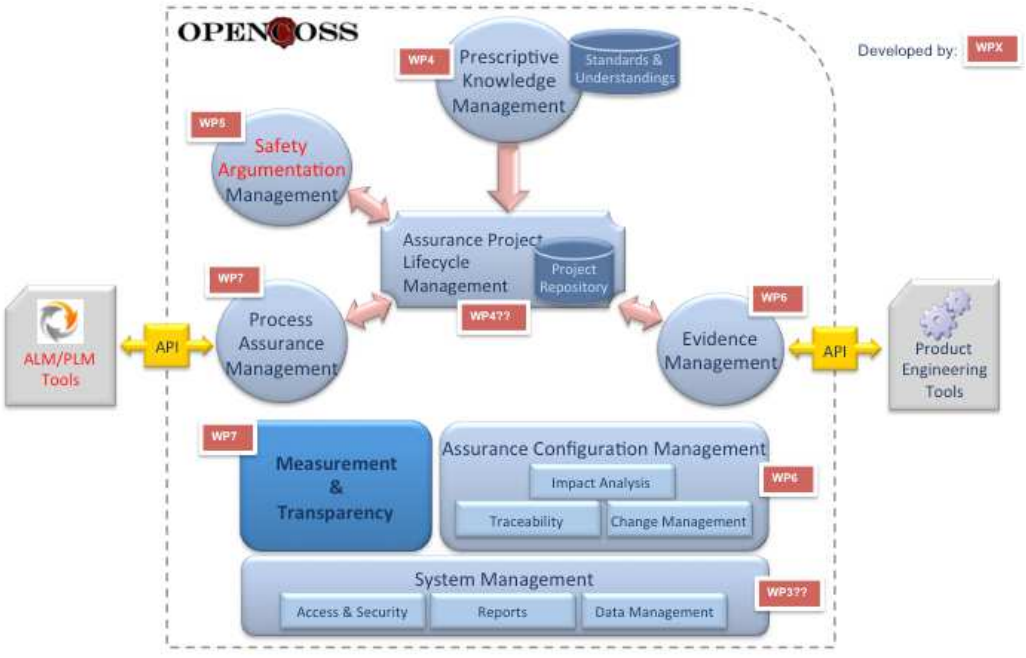

Figure 6: OPENCOSS draft architecture

Only sections 3.1-3.3 are presented, as the requirements in section 3.4 concern how the research is carried out, and constrain the solution, rather than being related to the actual implementation in the OPENCOSS platform.

Requirement Set	Requirement ID's
Reuse of existing argumentation	01-01, The OPENCOSS Approach shall provide a consistent and constrained means for the expression of safety argument claims 01-02, The OPENCOSS Approach shall provide a consistent and constrained means for the expression of safety argument contracts 01-03, The OPENCOSS Approach shall provide a consistent and constrained means for the expression of contextual information used in safety arguments 01-04, The OPENCOSS Approach shall provide a consistent and constrained means for the expression of assumptions used in safety arguments 01-05 The OPENCOSS Approach shall develop a library of reusable pattern-based structural templates for modular arguments 02-01 The OPENCOSS Approach shall provide consistent and constrained means for the expression, and identification, of public goals 02-02 The OPENCOSS Approach shall provide consistent and constrained means for the expression, and identification, of 'away' goals 02-03 The OPENCOSS Approach shall provide consistent and constrained means for the expression, and identification, of evidence characteristics
Specification of Assumptions	01-03, The OPENCOSS Approach shall provide a consistent and constrained means for the expression of contextual information used in safety arguments 02-02, The OPENCOSS Approach shall provide a consistent and constrained means for the expression, and identification, of 'away' goals 03-03, The OPENCOSS Approach shall provide a means for the overall assessment of confidence within the composed argument
Validation	03-01, The OPENCOSS Approach shall provide a means for the overall assessment of evidence supporting the argument 03-02, The OPENCOSS Approach shall provide a means for the overall assessment of risk represented by the composed argument 03-03, The OPENCOSS Approach shall provide a means for the overall assessment of confidence within the composed argument
Gap Analysis	01-06, The OPENCOSS Approach shall provide a means for managing change within a modularised argument 03-01, The OPENCOSS Approach shall provide a means for the overall assessment of evidence supporting the argument 03-02, The OPENCOSS Approach shall provide a means for the overall assessment of risk represented by the composed argument 03-03, The OPENCOSS Approach shall provide a means for the overall assessment of confidence within the composed argument

Table 2: Requirements against alternative viewpoints from WP2

4 Conclusions

This document presents (research) requirements for development of techniques to promote and exploit the harmonisation and reuse of the logical reasoning, which is used to provide assurance of the safety of components, subsystems and systems and of the evidence.

Claims within a safety argument are developed in natural language. Constraining the way these are expressed will help in the comparison between them. Arguments include the evidence quality/clarity and our confidence in it, and that a component really does what it is specified to do.

Challenges with component arguments for modules are:

1) To identify which claims to represent as public goals, i.e. which should be offered for other argument modules. If we offer too many goals, then the interface becomes difficult to manage and there will be too many safety case contracts required. However, subtle details may be missed if the public goals are at too high a level within the argument, and too broad in their scope.
2) To look at broad similar types of component (e.g. software, programmable logic, interlocks), and see if patterns for information that should be in the argument module can be provided. This will again help with the composition of arguments, ensuring matching of relevant claims about similar properties e.g. failure rates.
3) To ensure that assumptions and contextual information about the component, and the environment in which evidence was gathered and analysis performed, is expressed clearly and unambiguously.

The argumentation needs to be written in a way that allows translation within all elements, particularly claims. GSN (with module extensions) is proposed as a way to structure argumentation with explicit expression of module arguments. Also, WP5 needs to be able to identify where translation is not possible and where there are missing or mismatched assertions from one domain to another.

The requirements are grouped into four areas for further research:

5) High-level compositional argument methods
6) Low-level detail on how to develop safety argument modules
7) Managing and analysing emergent properties within the argument
8) Relation to more general project requirements, for implementation and relationships to other work packages.

This further research will occur in the next WP5 task; development of an approach for characterizing argument modules and evidence as well as the contracts between safety case modules. This conceptual framework will cover different strategies for safety case decomposition into modules, identifying common “types” of relationships between the modules and corresponding scenarios for composition as well as, if appropriate, corresponding templates for safety case contracts. Also, an overall framework for composition of safety cases from modules along with any identified limitations will be presented. It will contain a final set of prioritized requirements for WP5 tool support.

5 References

[1] OPENCOSS Project, *Business Cases and user needs, OPENCOSS Deliverable 2.1*, June 2012.

[2] OPENCOSS Project, *High-level Requirements, OPENCOSS Deliverable 2.2, August 2012*

[3] OPENCOSS Project, *Platform Architecture, OPENCOSS Deliverable 2.3, work in progress*

[4] Goal Structuring Notation Working Group, O. C. (2011, November 1). *GSN Community Standard* . Retrieved from http://www.goalstructuringnotation.info

[5] Meyer, B., *Applying 'Design by Contract'*, Computer, Vol 25: Issue 10, Oct 1992.

[6] OPENCOSS *Project, Baseline for Common Certification Language, OPENCOSS Deliverable 4.1*, April 2012.

[7] OPENCOSS Project, *Baseline for Compositional Certification Approach, OPENCOSS Deliverable 5.1*, July 2012.

[8] Fenn, J., & al, e. (2007). *Safety Case Composition Using Contracts - Refinements based on Feedback from an Industrial Case Study*. 15th Safety Critical Systems Symposium. Springer.

[9] Kelly, T.P., *Arguing Safety - A Systematic Approach to Managing Safety Cases*, DPhil. University of York, 1999.

6 Abbrevations

CCL	Common Certification Language
COTS	Commercial Of-The-Shelve
GSN	Goal Structuring Notation
Dx.y	Deliverable x.y (in OPENCOSS project)
MBA	Model-Based Engineering
OPENCOSS	Open Platform for EvolutioNairy Certification Of Safety Systems
V&V	Verification and Validation
WP	Work Package