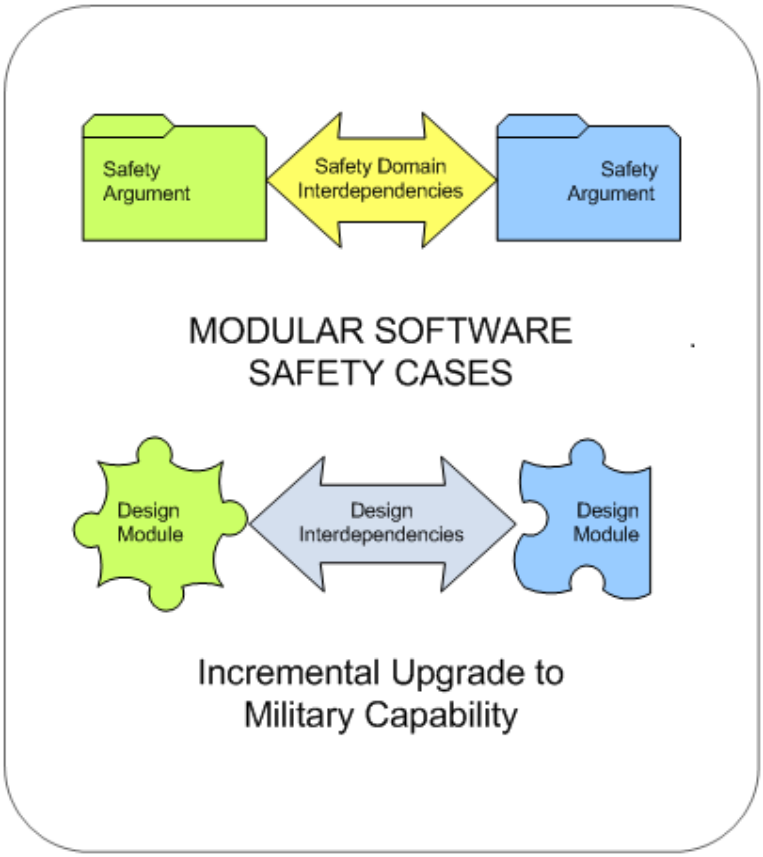
Modular Software Safety Case Process Overview

Date: 19-Nov-2012



Contents

Figures

1 Introduction

1.1 Background

The subject of this document is the *Modular Software Safety Case* (MSSC) process developed over several years through collaboration between five member companies of the Industrial Avionics Working Group (IAWG), with the support of Dstl.

It is recognised the trends of the growing size, dynamism and complexity of systems in increasingly variable operational and threat environments present significant challenges to the efficient provision of supporting Safety Cases. The development and use of the MSSC process is a response to these challenges.

The process was defined by a team of highly experienced engineers, expert in software development and safety assurance. It has been refined through experience gained from large scale trial applications of the prototype process, and further trials of the refined process.

A large scale trial was conducted on software hosted on a multi-layer computer architecture known as *ASAAC* (Allied Standards Avionics Architecture Council)[1], which presented specific challenges in making arguments about safety. These were addressed through specific patterns of argument developed within the generic framework of the MSSC process. Where appropriate, lessons learned were used to refine the generic MSSC process. However IAWG partner companies retain specific experience, knowledge and resources (e.g. argument patterns) relevant to ASAAC and to layered architectures generally.

The MSSC process has further benefited from the feedback of independent safety advisors, and from review by engineers from Dstl and other organisations.

The MSSC process is generic in nature, and able to be applied to Software systems of any architecture. It is not tied to any particular standard, such as DO178B or DEF STAN 00-56, and may be applied whilst conforming to these and other standards.

The MSSC process is provided to guide the development of Modular Safety Cases for complex software in systems which may have safety-related requirements requiring assurance to a single or, potentially, multiple assurance levels.

[1] ASAAC – A consortium, formed by European government and industry representatives (UK, France, Germany) which developed and gave its name to an Integrated Modular Avionics System Architecture.

1.2 What is a Modular Software Safety Case?

A ***Safety Case*** (SC) may be characterised 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 operating environment.***[1]

A MSSC differs from a traditional "monolithic" software Safety Case in that it is structured as an integrated set of carefully chosen modules relating to different aspects and partitions of the software system and its supporting hardware and architecture.

The MSSC process is designed to enable the development of a Safety Case in which the modules are as far as possible self-contained - that is they have low coupling to other SC modules. This allows a MSSC to be more easily, and cost-effectively, updated throughout the life of the system to which it applies, lowering an important barrier to the incremental modification and gradual through-life evolution of a complex system. Furthermore application of the MSSC process encourages the development and publication of Safety Case modules which can be re-used in subsequent systems which have common aspects of design or use.

Although the MSSC process was developed in the context of sophisticated military aircraft avionics systems, the generality of the approach makes it equally applicable to complex systems in other fields.

1.3 Purpose and Scope of This Document

This document aims to provide an accessible introduction to modular Safety Cases and the MSSC process. It introduces the issues which need consideration during the application of the process to a programme, some justification of why it's a good process to use, what benefits it brings and under what conditions it should be used.

Its relationship with other documents is described in Section 1.4 below.

This document sets out to answer a number of questions:

- *what is MSSC?* - Section 1 introduces the process, which is further elaborated throughout the document
- *why use MSSC?* - Section 2 describes what motivates the development and use of the process
- *when is it appropriate to use MSSC?* - Section 3 describes factors to consider in deciding whether MSSC should be used on a particular project
- *how is MSSC performed?* - Section 5 provides an overview of the process and Section 6 and 7 provide further detail
- *where does MSSC fit in?* - Section 9 sets the process in context with other related activities

Section 1 contains introductory material, including this subsection.

Section 2 lays out the case for adoption of the MSSC process by explaining its purpose, objectives and benefits.

[1] Note: Safety cases which provide evidence prescribed by standards rather than a structured argument defer to the standards authority for assurance that the evidence is sufficient and has the strength to support the claims of safety.

Section 3 discusses factors involved in the “Adoption Argument" – the rationale for choosing to use the MSSC process.

Section 4 introduces the description of the MSSC process by initially setting out some of the principles and terminology of Safety Cases.

Section 5 provides a brief overview of the MSSC process steps.

Section 6 introduces the main elements used in constructing a Modular Safety Case and describes these in detail by referencing appropriate documents. Key terms are defined in the Glossary [202]

Section 7 then explains how those elements can be used to create an argument framework.

Section 8 gives a summary of the management activities associated with the MSSC process.

Section 9 sets the MSSC process in the context of the wider Systems Engineering and Safety Engineering activities.

1.4 Relation to Other Process Documents

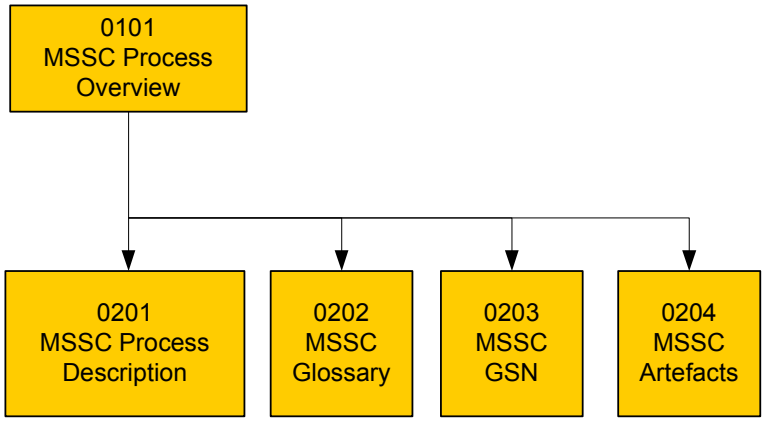


Figure 1: Structure of MSSC Process Document Suite

This document forms part of a two-level document suite, shown in Figure 1: Structure of MSSC Process Document Suite, which introduces and defines the MSSC process. This document is the only ***Level 1*** document***.***

The ***Level 2 documents*** ([201][202][203][204]) define the MSSC process, describing the associated artefacts in detail. This material is relevant to all software and processor platform architectures.

This document provides an overview of the process, and contains introductory material which omits the full detail. Should any apparent differences exist the detailed process and artefact definitions in Level 2 documents take precedence over the simplified introductory material in this document.

1.5 References and Further Reading

[201] MSSC 201 – MSSC Process Description, Current Issue Applies

[202] MSSC 202 – MSSC Glossary, Current Issue Applies

[203] MSSC 203 – GSN (Summary), Current Issue Applies

[204] MSSC 204 – Artefacts, Current Issue Applies

2 The Benefits of MSSC

2.1 Modularity and its Benefits Generally

Systems and software have been designed according to modular principles for many years. The modular approach means the partitioning of a system into an inter-connected set of subsystems, defined by the characteristics they present *at their boundaries*. This allows analysis of the system to be performed by considering the interaction of subsystems - each with its given (boundary) characteristics - as a function of the way they are interconnected. Thus analysis *at system* level does not require analysis of the internal behaviour of any subsystem.

The problem of ensuring that the design of each subsystem meets its agreed boundary characteristics is critical to system verification, but is conducted as a separate activity from system-level analysis. It can be approached in the same, modular, way as system level design and verification, i.e. in terms of an interconnected set of “sub-subsystems”.

There are many benefits to adopting a modular approach in system or software design. In particular the resulting independence of analysis between different levels of the system hierarchy allows:

- the simplification of analysis resulting from the reduction in complexity of dealing with only one layer in the system hierarchy at a time
- the delegation of different aspects of the overall design to teams with specific domain knowledge related to the (sub) system in question
- the Intellectual Property relating to a subsystem’s internal design to be protected
- parallel working at different levels of system design and implementation
- the use (or re-use) of identical modules in more than one system

Also, ideally, any subsystem may be replaced by another with the same (critical subset of) boundary characteristics without impacting on the established characteristics of the system. This allows for the management of obsolescence, the incremental introduction of new technology and promotes competitive procurement.

This modular approach requires a disciplined design process to ensure subsystem boundary characteristics are well chosen, precisely defined, agreed as feasible by the subsystem designer and are subsequently met and verified.

2.2 The Benefits of Using a Modular Safety Case

Just as there are benefits to be gained in modularising the architecture of a system, it is possible to gain similar benefits in modularising the architecture of the Safety Case and safety arguments relating to a system. Indeed allowing the Safety Case modularity to be clearly aligned with that of the system architecture also provides the prospect of improved maintainability of the system itself.

The MSSC process requires Safety Case Module boundaries to be defined, and self-contained argument patterns to be defined within those boundaries. This has a number of benefits, including those identified below:

2.2.1 Facilitation of Incremental Change and Evolving System Capability

A well partitioned Safety Case limits the impact of incremental changes to a minimal area of the Safety Case. Thus any incremental change should be made more quickly and with less effort. This ensures that the maximum amount of Safety Case material is available for re-use without it needing to be revisited.

Thus an important advantage of a modular Safety Case is to reduce the effort and time needed to update the Safety Case of a system following a change which potentially affects safety.

The saving in time and effort for incremental upgrades may be exploited by incorporating a series of incremental changes over the lifetime of a system. The evolutionary development of both the system and the Safety Case means that system changes do not need to be "saved up" for a major update.

A key aim of the MSSC process is to enable system changes to be applied as a series of relatively small increments, rather than occasional major updates, maintaining the system at, or close to, peak operational capability at all times as its requirements evolve.

2.2.2 Re-use of Safety Case Modules

The specification of common modules of software and hardware for use across a range of systems is an important strategy for reducing the procurement and maintenance costs of defence equipment. The adoption of a MSSC allows Safety Case Modules developed for one system to be re-used for different systems where corresponding system components are reused and where the safety requirements and context are the same or very similar.

As the adoption of common system modules increases in future, as intended by the defence procurement authorities, the cost of generating a Modular Software Safety Case for a new system is expected to be correspondingly reduced.

An important aspect of the Re-use of Safety Case modules is that it allows the experience and knowledge of safety engineers and system designers to be captured, reviewed and developed over the long term, as sophisticated argument patterns are generated, validated and expanded over several generations of system.

2.2.3 Advantages to Safety Case Development

Use of a MSSC brings a number of important benefits in Safety Case development, including:

- Ease of construction (work sharing) and managing scale in complex and/or larger systems.
- Safety case modules can be authored independently
- Authors can own their Safety Case module IPR
- Safety cases for multiple system variants are handled more easily
- MSSC can be applied efficiently to mixed assurance level systems
- Provides focus on integration boundaries
- Provides expert patterns for standard arguments
- Allows earlier development of the Safety Case

3 Suitability of Systems for MSSC

Before committing to adopt modular Safety Cases as opposed to traditional “monolithic” Safety Cases it is recommended, that a preliminary analysis of the expected benefits, costs and risks (that is a cost benefit analysis) is performed and maintained as an “Adoption Argument” document. This collects together and presents the information which forms the basis of the decision, sets out the decision rationale and can aid future management and planning of the programme.

It is also recommended that the approach to certification needed to realise the full benefits of a modular Safety Case be agreed with the project specific certification authority.

The development of a MSSC includes a number of steps over and above those traditionally taken in generating a monolithic Safety Case. This may involve additional time, effort and cost, especially where previously developed MSSC modules or patterns are not available for the majority of the system.

The return on this up-front investment is the containment of predicted future changes, and the expected return on investment is dependent on the expected degree of change which the system will be subjected to in the remainder of its service life.

Thus the Adoption Argument should take account of the anticipated lifecycle of the system.

Significant factors include the anticipated length of service of the system, planned changes, and the potential for unplanned system changes in terms of frequency and scope.

Past experience and the examination of the actual lifecycles of previous systems are likely to provide valuable source material. This information may be collected together in a Project Lifecycle Plan (PLP).

The MSSC process may be followed concurrently with the design of a new system. This allows the design process to be influenced by the rigorous analysis of safety arguments, resulting in an improved system design from the safety engineering perspective. However it may also be feasible to apply the MSSC process retrospectively to a legacy system of settled design for which a conventional (non-modularised) but compelling Safety Case already exists. In this case the process allows modules to be defined within the Safety Case which bound the impact of software changes on the Safety Case argument in order to reduce the complexity and effort involved in updating a Safety Case for re-certification.

There are many factors which should be considered in assessing the practicality and value of adopting the MSSC process for a specific project and system.

MSSC is most beneficial in managing change in large and/or complex systems, and delivers greatest benefit where there are expected to be a number of small to medium size changes implemented throughout the life of the system. It is simpler to introduce for systems that use modular design approaches and supports the re-use of design modules that have an existing safety argument.

Appendix A *“Receptivity Analysis”* provides a set of structured questions which should be considered in assessing the suitability of using MSSC on a particular project.

4 Principles of Safety Cases

In general, the term “Safety Case” refers to all the documentation which contains the Safety Case Argument and the evidence.

The method adopted to construct the Safety Case divides a Safety Case into 3 main parts.

(1) ***The Claim*** - e.g. that the system is adequately safe. This usually has some context - e.g. to fly in civil airspace only.
(2) ***The Argument -*** This sets out why we believe that the claim is true - e.g. in order to be safe it must behave like this; I claim it does so and here's the evidence.
(3) ***The Evidence*** - e.g. quality stamped test results or analysis.

In a realistic situation the argument may consist of many sub-arguments and pieces of evidence. In all cases they are attempting to assure a certifying authority that the claims are valid, requiring that sub-claims provide effective support to them.

Similarly the evidence needs to be

- relevant to the supported claim,
- from a range of diverse sources,
- able to provide sufficient assurance to the supported claim,
- valid and accurate.

The ***argument*** connects the ***claim*** with the ***evidence*** that supports it. The purpose of the argument is to make this connection in a compelling, comprehensible and valid way.

For each step in the argument the totality of the supporting claims should both

a) be logically equivalent to the supported claim

b) cover the same subject and scope of subject as the supported claim.

The type of evidence is related to the strength of the supported claim. A stronger claim usually has more or more varied evidence supporting it.

In a top-down approach, for example, a claim may require only test evidence. Should the context require a higher level of assurance, however, then evidence of peer review of the tests or analysis may also be required and may need to be referenced by the argument.

In a real situation the form of the argument selected may depend on the type of evidence which is available or can realistically be collected. Nevertheless the argument and supporting evidence must provide sufficient assurance of the validity of the claim.

The method recommended[1] for capturing this information within the MSSC process is to use Goal Structuring Notation (see [203]), where **Claims** are placed in box shaped diagrammatic elements known as "**Goals**", and the Argument is captured in diagrammatic relationships to **sub goals**. These are qualified by context information, including justifications and assumptions, which are captured in rounded box and elliptical diagrammatic elements respectively. The **evidence** is referred to from within circular diagrammatic elements called **"solutions**".

[1] This does not preclude the use of other notations (e.g. CAE or plain text). However, MSSC has only been demonstrated using GSN and care must be exercised that, if using other notations, they are capable of expressing the modular safety case as completely and comprehensibly as required, as is the case with GSN.

5 Overview of the MSSC Process

The MSSC process splits the development of a new modular Safety Case into four steps. These are represented in Figure 2 along the upper path. The lower path shows the steps to be undertaken to accommodate change during the subsequent application of the Incremental Safety Case process.

Note: the Incremental Safety Case process can only be successfully used where the original Safety Case has been developed using the MSSC process. Thus Modular Safety Cases need to be constructed for legacy systems (via the upper path) before incremental Safety Case process can be applied to them.

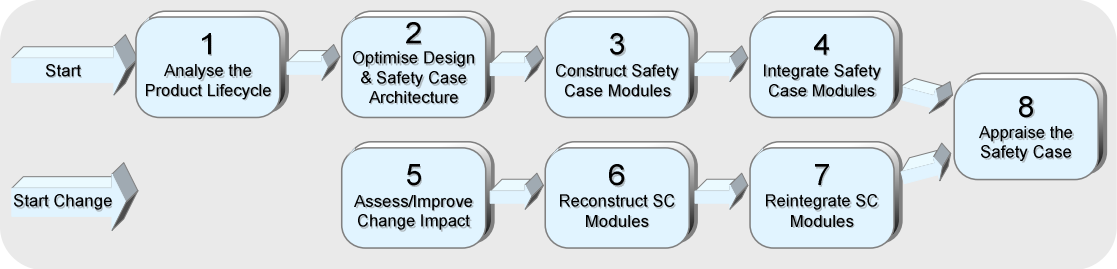


Figure 2: Modular and Incremental Safety Case Process

5.1 Purpose of MSSC

The Industrial Avionics Working Group has developed the Modular Software Safety Case (MSSC) Process which supports the construction of a Safety Case by combining Safety Case modules. The MSSC process is designed with the aim that a Safety Case developed in accordance with the process exhibits the attributes required of a Safety Case, as follows:

- *Structured*; a Safety Case produced according to the MSSC process is structured such that safety claims, arguments and evidence relating to defined partitions of the system are contained in distinct, coherent Safety Case modules.
- *Supported*; all MSSC claims must be supported by evidence. Application of MSSC encourages evidentiary support to be optimally organised between SC modules.
- *Compelling*; The argument supporting a claim, and the evidence it relies on, must be clear, unambiguous and complete, leaving no room for doubt that the claims of the Safety Case are valid. The specifics of the argument remain the responsibility of the design authority. However, in appropriate circumstances, argument patterns generated using the MSSC process may be re-used.
- *Comprehensible and Valid*; The SC must be clearly presented, such that it is able to be understood and scrutinised at all levels of detail by those involved in its construction and assessment, and in the engineering and use of the system to which it applies. The MSSC process uses Goal Structuring Notation (GSN). GSN provides a clear means of navigating through an argument. Scrutiny and analysis of the argument is simplified by its division into modules that are focused on parts of the product and the process.
- *For a given application and environment*; Safety Case claims generally apply to a defined set of circumstances, including specified environmental conditions and operational practices. Arguments and evidence may also be valid only under specific environmental and operational conditions. These (along with other safety relevant details not expressed in the argument claims) must be captured by MSSC as GSN Context. GSN Context must be sufficiently complete and compatible across all the component parts of an argument supporting a claim for it to be valid.

In addition the MSSC Process is designed to maximise the following attributes:

- Time to Market; A MSSC SC Architecture facilitates developing individual SC modules early and prior to SC integration, helping to bring SC effort forward.
- Flexibility; The SC modules are chosen so that the impact of a change to the SC will be contained by them, making the SC easier to change.
- Cost Effectiveness; The investment in the SC modularisation made by MSSC is directed to achieve a return over the anticipated system lifetime.

A Safety Case is used in the process of "Certification" – i.e. a safety case provides formal assurance that a system is safe for a given application in a given operating environment.

The MSSC process addresses the development of a Modular Safety Case. Such a Safety Case comprises a set of independent Safety Case Modules within a "Safety Case Architecture" to form a coherent Safety Case for the system.

Following a change to a previously certified system, for which a modular Safety Case exists, an impact analysis may be conducted to identify those Safety Case modules which need to be changed. Where a well-structured Modular Safety Case exists, incremental changes to the system may result in modification to only one, or a small number of Safety Case modules. It is an aim of the MSSC process to structure the Modular Safety Case such that it minimises the impact on the Safety Case of changes which might be expected during the life of the system making the system more flexible and able to accept changes.

The process of updating a Modular Safety Case following a system change is known in this document as the *Incremental Safety Case* process. This is illustrated in

Figure 3: An Incremental Safety Case Update Limits the Impact of Change

, which shows how a limited change to a system (the replacement of a software module – or "Block") has a correspondingly limited impact on the Safety Case.

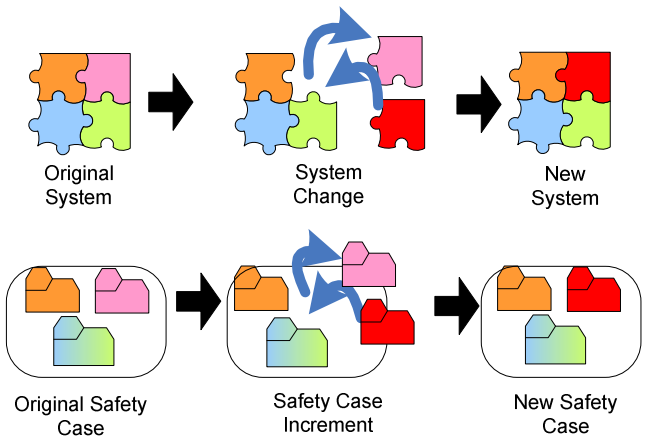


Figure 3: An Incremental Safety Case Update Limits the Impact of Change

When the architecture of a Modular Software Safety Case is defined, the expected life cycle of the product is analysed to identify reasons why the design may be changed. These could relate to different circumstances such as technology upgrades, requirement changes,

obsolescence or export opportunities. In MSSC these circumstances are known as “Change scenarios”.

Analysis of change scenarios allows the Safety Case Architecture to be structured to limit (or “contain” the impact of anticipated changes to one or a small number of SC Modules. The MSSC incremental Safety Case process is intended to generate a clear and specific argument for submission to the certifier that they do not need to re-visit unchanged or re-used parts of the system, along with a compelling argument that the system remains safe in respect of the changes.

During the service life of a system, as technologies and operational requirements evolve, a series of incremental changes may be applied to maintain the system at full operational capability. The MSSC process is designed to allow the Safety Case to be updated for each of these changes in a safe and efficient manner. Therefore, as the system evolves over its service life, the Safety Case evolves with it.

This evolutionary development of both the system and the Safety Case contrasts with previous practice in which system changes are often “saved up” for a major update, such as an aircraft “Mid-Life Upgrade”. It allows capability to be maintained close to its full requirement, as illustrated in Figure 4.

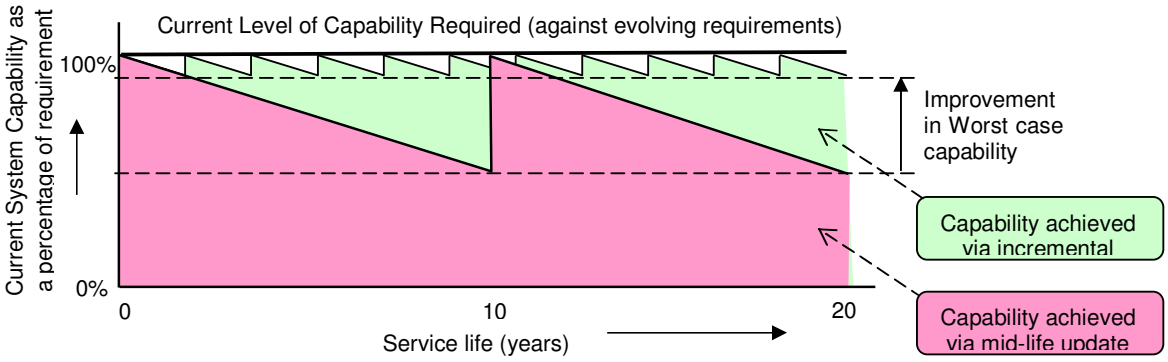


Figure 4: Maintaining capability through life

Note that the focus of the Modular Software Safety Case Process is on the safety aspects of the software subsystem. It does not address other system aspects such as functionality or performance, except where they are both relevant to system safety and lead to software safety requirements.

Note also that if the environment in which a system is required to operate changes (e.g. when the role of an aircraft is re-designated from Air Superiority to Ground Attack) then the Safety Case must also change. That is the Safety Case applies to a *defined application and environment.*

5.2 Development of a New MSSC

A new MSSC may be developed concurrently with the design of a new system, or may be developed shortly after system design. Note that a MSSC may also be developed for a legacy system, of settled design, in order to allow for subsequent application of the Incremental Safety Case process. In any case the process follows five numbered steps as shown in Figure 2:

The process steps are described in the following paragraphs. For clarity and ease of reference each step is displayed in its own outlined box.

Process Step 1

Analyse the Product Lifecycle

The first step of the MSSC is to analyse the expected full Product Lifecycle (typically over 25 years) in order to establish a set of the most significant potential change scenarios. This set of Change scenarios is used to predict the impact on the Software Design of changes which might arise from different circumstances.

This analysis is critical to understanding the potential benefits to be gained from applying the MSSC Process. It informs the structuring of the modular Safety Case architecture. Important issues considered are:

- the longevity of the product
- the range applications to which it is expected to be applied in that time
- how often it is likely to be changed over its life?
- what is the likely nature of the changes ?
- how localised are the changes are likely to be ?

Process Step 2

Optimise Design & Safety Case Architecture

The full set of Safety Case Modules include those relating to coherent parts of the Application software known as Blocks in MSSC, and their interactions, as well as those relating to other aspects of design and process upon which the Safety Case depends.

The Safety Case Architecture (SCA) identifies all the SC modules and the shows the way in which these will be integrated to form a structured argument that the software satisfies its safety requirements.

Within Step 2 the SCA and Software Design are examined and revised in the light of the change scenarios identified in Step 1, in order to minimise the potential impact of expected changes. Also within this step the engineering team identifies and assesses candidate Software Designs and possible alternatives for parts of the SCA, and establishes mutually optimised solutions for both.

The optimisation of these architectures is an iterative process. It begins with the definition of a number of design and SCA alternatives based on modularity that exists in areas such as the system, evidence or processes. These architectural alternatives are evaluated across the set of change scenarios established in Step 1, and refined and improved as appropriate. This step is iterated until an acceptable balance between the objectives of the design architecture and those of the Safety Case architecture is achieved.

The considerations for constructing architectural alternatives are;

1) Process based divisions.
2) The containment of, and susceptibility to change.
3) The relationship between the SCA and Design Modularity.
4) The need to have distinct independent sets of evidence supporting the argument in each SC module (evidence modularity).
5) Architectural Patterns.

There are a number of process-based influences on the choice of modules which should be considered including those listed below:

- Different required levels of assurance - e.g. SIL1 or SIL 2 (which define the required strength of assurance).
- Different responsible, regulatory or legislative organisations (i.e. differing standards)
- Different implementation teams
- The distinction between processes for legacy assets and new components
- The distinction between processes for COTS and bespoke components.

Consideration of future changes enables the separation of elements likely to be subject to a high frequency of change from those with a low frequency of change.

The considerations for evaluating the architectural alternatives against each other are;

6) Estimated magnitude of change containment benefits against investment
7) Whether they assist with Complexity Management
8) Whether scrutiny should be focused upon a Design Module boundary to improve safety.

The public interfaces between SC modules, including claims and context, are defined by the SC Architect at an appropriate level of detail for the programme. Ownership of these public interfaces passes to the owners of the individual SC modules, but they remain a part of the architecture.

The output of step 2 may be used as part of the preliminary SC, along with any argument strategies within SC modules that would benefit from early scrutiny.

Process Step 3

Construct Safety Case Modules

The objective of this, and the following step, is to provide a compelling and comprehensible Modular Safety Case which ultimately is supported by a body of evidence that is sufficient to demonstrate that the software system satisfies its safety requirements.

At this point in the process the Safety Case Modules need to be populated with the rest of their externally visible ("public") claims and all the internal claims and arguments which support them.

In addition to modules that argue about the safety-related guarantees of functional elements of the system, there will be other modules supporting different aspects of the Safety Case, such as any behaviour or properties that cannot be attributed to a single Block or argued about by a Block related SC module.

For example each one of a set of functional blocks within a software system may place its own demands on a shared resource, such as computer processing load. Safety Case Modules may be required to provide a compelling argument that the aggregated demands of the system will be met by the processing platform. Similarly Safety Case Modules may be required which argue the case for non-interference between systems elements and the independence of their corresponding Safety Case modules.

Another aspect which must be considered is the distinction between arguments about the product itself and arguments about the process by which the product is developed.

In addition to arguments the Safety Case Modules provide links to evidence appropriate to the argument.

The types of evidence which may be used include the results of unit test, qualification test, static analysis, design features and review etc.

The validity of evidence, such as test results or analysis, in supporting a claim can be limited by the operational and environmental context in which it is obtained, or to which it applies. Therefore specific conditions and assumptions pertaining to testing and analysis must be identified or elicited and recorded, as context for the SC claims along with the results, for subsequent analysis.

Process Step 4

Integrate Safety Case Modules

Ultimately the aim of the Safety Case is to provide assurance that the system which is its subject is adequately safe when used in a defined environment and operational context. (What constitutes "adequacy" has a project specific definition, possibly making reference to stakeholders, an assumption that stakeholders have completed scrutiny of the SC, and/or any standards they prescribe.)

The individual Safety Case Modules need to be integrated into a complete Safety Case that provides a compelling argument that compliance with the system safety requirements is assured. A key step in this is to link the claims which require support, to those claims which support them, and ultimately lead to the evidence that underpins the Safety Case.

The *context* associated with Safety Case claims capture assumptions, constraints and definitions associated with those claims. It is important that the context associated with goals shared between Safety Case Modules is compatible, not least where the Safety Case Modules are developed independently by different developers,

Typically the output of step 4 would be used as the interim SC.

Process Step 8

Appraise the Safety Case

The purpose of undertaking this appraisal is to evaluate the completed Safety Case against its initial set of objectives and desirable attributes, in particular to verify its present quality and its likely effectiveness throughout its expected lifetime.

The appraisal of the Safety Case has two aspects:

The first provides the SC developer with an opportunity to step back and consider whether the Safety Case is adequate. This aspect should involve Independent Technical Authorities who have special responsibilities for safety or who will be ultimately signing off the system.

The second aspect is to appraise the efficacy of applying the MSSC process in producing it. This provides the developer with the opportunity to review whether the Modular Safety Case remains aligned with the anticipated product lifecycle and whether, as a result of experience gained in preparing the current Modular Safety Case, there are opportunities to improve the processes used in subsequent development.

The objectives of the MSSC to be assessed are:

(1) Robustness - the argument needs to be compelling and defendable. This also implies clarity and understandability as required attributes.

(2) Evaluation of the ability of the Modular Safety Case to cope with changes – in facilitating change and updates (from minor to major) to be incorporated allowing use of the Incremental Safety Case process.

It may be dependent upon the programme but typically the output of this step would be submitted for approval as the final SC. The level of detail and rigour of the appraisal performed varies according to whether the Safety Case is preliminary, interim, final or operational.

5.3 Safety Case Update for an Incremental Change

The process of incrementing a Safety Case updates a modular Safety Case for a previously certified system when the system is changed. It allows the re-use of Safety Case modules from the previous system certification. A key advantage over the traditional approach is that upgrades can be smaller and made more frequently, allowing the system to evolve and maintain or increase capability over its operational life.

Referring to Figure 2, the process steps are as follows:

Process Step 5

Assess/Improve Change Impact

MSSC Impact analysis is a process by which the changes to a Safety Case are identified during an increment. The Safety Case team and design team will participate in this step and, in some cases, may call upon users or customers to take part. It may lead them to improvements to the system, the system lifecycle plan, the test plan or other elements. The objectives of this step are to identify what changes are necessary to the Safety Case and what parts of the existing Safety Case are not affected and therefore do not need to be revisited. It is a key benefit of the MSSC process to isolate updates to the Safety Case to where they are necessary, therefore saving time and effort.

In addition this part of the process is used to identify where beneficial modifications to the Safety Case architecture may be made and to identify beneficial modifications to the system or system lifecycle plan etc.

Process Steps 6 and 7

Reconstruct and Re-integrate Safety Case Modules

The processes set out in step 3 *Construct Modular Safety Case* are re-applied to the affected parts of the Modular Safety Case to construct the revised Safety Case. To re-emphasise the important point made in the previous section, the increment includes a Change Argument that rationalises why it is safe that un-affected parts of the Safety Case are not re-worked or re-visited. The identified changes are applied to produce a new Safety Case.

An important feature of this new Safety Case is that it stands alone - that is it replaces the old Safety Case, and is used at the starting point for any future updates. There is no need, or requirement to refer back to more than one previous version.

Process Step 8

Appraise the Safety Case

The Safety Case is re-appraised in a manner similar to that described in section 5.1

6 Principal Elements of a Modular Software Safety Case

The MSSC process is concerned with choosing boundaries within the software implementation (the *blocks)*, capturing the safety related aspects of their required behaviour and then developing the arguments contained in the Safety Case modules which correspond with, or relate to these blocks. For a new development, the emerging architecture of the Safety case may influence the software design architecture.

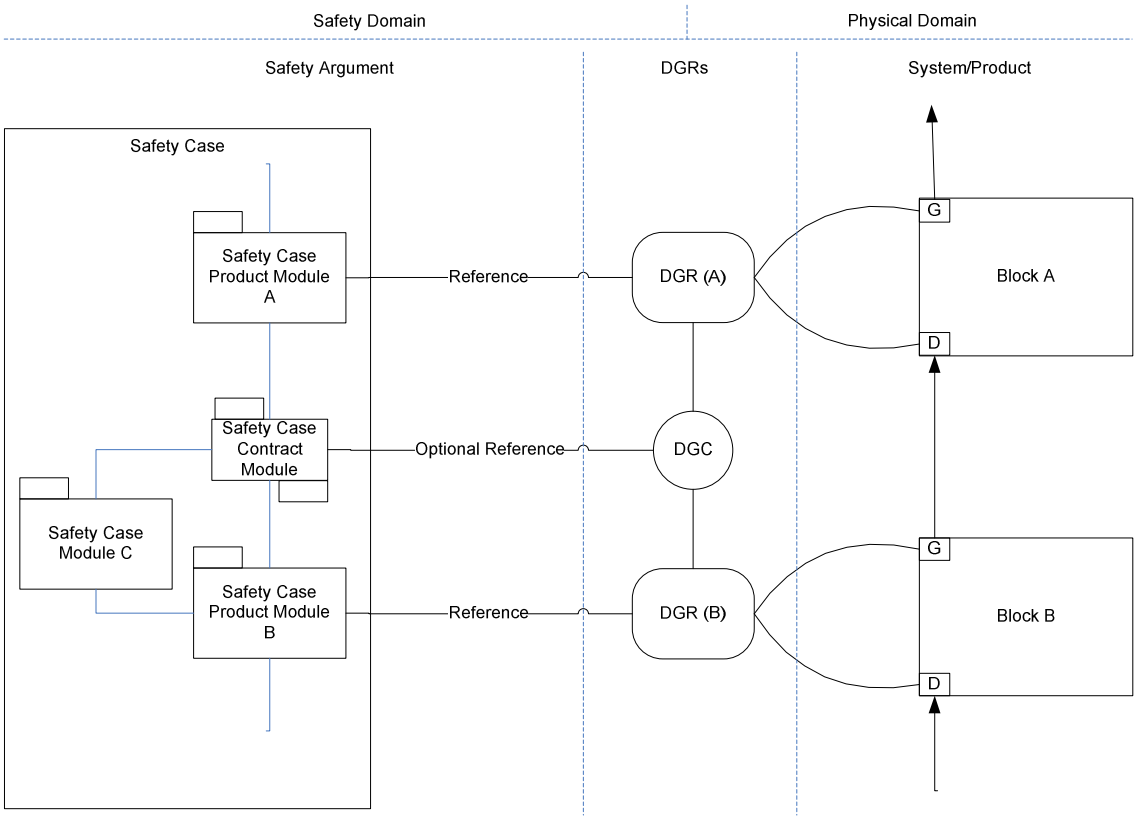


Figure 5: Principal Artefacts (Elements) of a MSSC relate to System/Product artefacts.

A Modular Software Safety Case typically includes a number of different types of artefact. Figure 5 illustrates these and shows how they might interact. The following is an overview of each type of artefact. More detail is provided in [204].

- **Safety Case Module**: A Safety Case Module contains one or more claims about all safety related aspect of the block of implementation, supported by evidence and an argument. It also contains information about the context of the claims.
- **Safety Case Contract Module**: A Safety Case Contract describes the links between a goal requiring support in one Safety Case Module with one or more goals providing support in other Safety Case Modules. It may also contain an argument for the sufficiency of the support.
- **Block:** An identifiable part (or group of parts) of the S/W implementation that is chosen by the SC Architect to be the subject of a '*Safety Case Module*.
- **Dependency Guarantee Relationship (DGR)**:- The DGR records the provisions of a block (the guarantees)(G) that are relevant to the Safety Case, and the requisites on which it depends (the dependencies)(D). This information forms the basis of the

product argument for the block.
This recognises that the safe operation of a block may depend on the safe operation of other blocks. The fact that certain guarantees are related (only) to certain other dependencies is described in a Dependency Guarantee Relationship (DGR). The DGR captures the “D” and G” in a block that are associated with each other..

- **Dependency Guarantee Contract (DGC)**:- A DGC records the relationship between Blocks, for example that the dependencies of Block A are supported by associated guarantees from Block B. Note in practice several Blocks, and several DGCs, may be involved in providing that support. DGCs may be useful to providing context for the SC contract argument module.

A Modular Safety Case (SC) is composed of SC Modules which contain the claims, arguments, and evidence for each of the blocks. The individual Safety Case Modules can be linked together directly, or using Safety Case Contract Modules to form the complete MSSC (as is the case in Figure 5).

In Figure 5 Blocks A and B, and their corresponding SC Modules, refer to non-overlapping functional partitions within the system.

It is possible, in the correct circumstances, to alter one Safety Case module without impacting another, as explained below.

Safety Case Module A argues that Block A operates to uphold its guarantees provided its dependencies are met – i.e. it claims, argues and provides evidence that the guarantee provided by Block A , as defined in its associated DGR is sufficiently assured. Safety Case Module B does the same for Block B.

Safety Case Module A and Safety Case Module B do not argue directly about each other. The Safety Case Contract Module captures the connection between the two SC Modules to show how the overall SC depends on the assurance of both Block A and Block B.

In this way Module B argument is permitted to be modified within the limits of the contract without affecting the argument in Module A.

However also shown in Figure 5 is SC Module C. SC Module C is equally critical to the Safety Case and must, for example, provide a convincing case that the platform on which the software runs (the “platform architecture”) is sufficiently assured that the software modules can operate according to their safety related requirements.

7 MSSC Safety Case Architecture

In a real system many argument threads are required to provide the necessary assurance that all safety related requirements of the system are met, and many SC Modules, their inter-relationships and pieces of evidence may be required to cover the various elements of the system.

A Safety Case Architecture:

- Identifies the SC Modules in the SC
- Describes each SC Module i.e. what it argues about
- Defines the support exchanged between the SC Modules' arguments.
- Describes how the top claims in the SC are intended to be used within the broader SC or certification.

For example in a layered software system these SC Modules may include those arguing about the behaviour of Application Layer software and those arguing about the computer architecture, including the operating system. Such a software system, based on the ASAAC architecture, has been the primary proving ground for the MSSC process.

8 Managerial Processes and MSSC

As with any development process, the activities leading to the generation of the Safety Case need to be integrated into the project management, and configuration management activity.

In particular the following aspects of project management have MSSC specific aspects:

8.1 Early consideration of SC issues

The Software architectural design can be significantly influenced by SC issues. It is important that arrangements are made to facilitate this occurring. For example a number of meetings involving both design and safety teams could be arranged to consider safety aspects of design, and the significant findings could be circulated more widely.

8.2 Manage Safety Case Report Iterations and Reviews

The MSSC process for a new development needs to be repeated at appropriate times in the software development and review lifecycle. So, for example, a draft Safety Case is normally produced for review early in the development before any actual evidence has been obtained.

This normally involves the production of preliminary, interim, final and operational Safety Cases. The Safety case report is reviewed at appropriate stages. In particular the adoption argument should also be reviewed to ensure that the business case remains valid (i.e. that costs are not spiralling as the Safety Case matures and becomes more stringent (e.g. from preliminary to interim SC).

This process needs to be managed and controlled to remain cost effective.

As the SC progresses from preliminary to operational status, the SC needs only to be sufficiently detailed to allow risk assessment at an appropriate level for the current stage of development.

Thus

- a *preliminary SC* may consist of the SCA along with strategies for any areas of argument considered risky.
- an *interim SC* may be the fully integrated SC prior to formal appraisal
- after approval (possibly as late as initial certification) the SC may reach *Final SC* status
- the operational Safety Cases that cover specific deployments of the system may later make use of the final SC.

8.3 Safety Case Module Team integration

The interrelationship of the SC Modules may need to be developed in collaboration between developers of two or more Safety Case modules also involving the SCA owner and their integrator. Once the SCA is defined, the SC modules may be developed in any order. It is best to address the major risks early if that is possible. This situation warrants particular management attention.

8.4 Metric Collection and Use

The benefits of modular Safety Cases, as detailed in Section 2.2 need to be measured in order to support prediction, control, and trade-off analysis.

9 The MSSC Process in the Context of Systems and Safety Engineering

The purpose of this section is to show how the Software Safety Case and its development relate to an overall system Safety Case, and to aspects of the wider systems engineering process.

It is important to note that the broader Systems Engineering and Systems Safety Assurance activities described here are outside the scope of the MSSC process, which focuses on the preparation of a Modular Software Safety Case.

9.1 System-level Hazard Analysis

System level hazard analysis is *not* part of the MSSC Process. However it is crucial that such an analysis has been thoroughly carried out for any system or subsystem Safety Case to be compelling. Hazard Analysis, its impact on Software Safety Requirements and the Software Safety case are addressed here.

Hazard analysis should be carried out at the beginning of the system development process, to ensure that the physical system is developed right from the start to mitigate inherent risk (e.g. with appropriate integrity levels). This will typically generate a set of safety related requirements for subsystems, including for software.

The MSSC process generates a software Safety Case which demonstrates in a compelling manner that all safety-related software requirements are satisfied, whether they are derived from the primary functions of the system, or in mitigation of system hazards.

Thus a system level Safety Case must assure readers that:

1. An adequate and sufficiently comprehensive assessment has been made of possible system level hazards i.e. a ***Hazard Analysis***
2. An adequate level of ***mitigation*** has been specified for all identified hazards.
3. There is sufficient ***evidence*** that the required mitigations have been implemented effectively.

The system design process may require repeated cycles of design, hazard analysis and rework, as the effectiveness of any mitigation is assessed, and the residual or emergent hazards associated with changed design are addressed.

Hazard Analysis

A functional analysis to identify potential hazards in the operational context of the system. It examines the way in which the system will be used, so that any dependencies on, or consequences of potential system behaviours are identified.

Hazards are initially addressed at system level, for example by engineering the design of the system so that these hazards are adequately mitigated, or by constraining the operational use or environment of the system to avoid or mitigate the hazards. Addressing the hazards at the top level will in turn lead to a number of related subsystem requirements.

As the system design progresses the nature of the solution may itself lead to the introduction of new potential hazards, for example through the use of components which may fail, or through more or less subtle interactions between elements of the design. For example the iteration frequency of a software thread may cause EM emissions at a frequency which interferes with safety critical hardware.

Various techniques may be used to identify such emergent hazards, including "bottom up" analyses of the impact of failures (e.g. FMECA), the use of hazard checklists, expert design reviews, brainstorming and test.

A suitable solution to address any system design requirement may involve placing a requirement upon software. Thus system safety requirements and design features introduced to mitigate hazards may give rise to **Software Safety Requirements**.

Example Software Safety Requirement

The system designer may elect to address the general system safety requirement ***"The pilot must be made aware if the aircraft speed is too low"*** by specifying that ***"the software shall command an audible warning when the aircraft Indicated Airspeed is less than 100 knots"***. He or she would also have to specify associated hardware and procedural requirements.

Note that, in general, as requirements are flowed down the requirement changes towards a more explicit definition of behaviour.

9.2 Safety in the Event of Failures

In any system there is a risk that components will fail and that these failures will impinge upon the ability of the system to satisfy its safety requirements. For example a software subsystem required to command a "low height" alarm will not do so if the processor it is running on fails. Equally it may not do so if a dormant software error prevents the specified behaviour of the software. Such failure modes must be addressed in the safety engineering of the system and in the overall System Safety Case.

The identification of software-related failure modes may be approached systematically by using a Software Failure Analysis technique such as SHARD.

Software Failure Analysis example: SHARD

` SHARD - a variant of the process industries' HAZOP technique, provides a structured approach to the identification of potentially hazardous behaviour in software systems. SHARD uses a set of guidewords to prompt the consideration of possible failure modes. Based on software failure classification research, five guidewords are used in the SHARD method - *omission, commission, early, late* and *value* failure. These guidewords are applied systematically to functions and/or flows in a software design.

For example, consider the function provision of "secure and timely data flow" to and from an application process. One possible failure mode prompted by the "omission" guideword could be that the data is not sent from the source. Use of SHARD facilitates the systematic identification of software contributions to system level hazards and the definition of associated software safety requirements.

The overall Safety Case (which includes the MSSC) must provide assurance relating to the following aspects of safety:-

(1) That the software system will meet its safety requirements (if any) under failure-free circumstances.
(2) That all credible failure mechanisms and circumstances under which the software system may fail to meet its safety requirements have been identified.
(3) That the remaining level of risk due to failures is compliant with System Safety requirements.

The MSSC is concerned with assurance of aspect (1) in all respects; and with assurance of aspects (2) and (3) at least in so far as that it must address and argue about the risk of software failures, and communicate these to the systems safety engineering process.

The precise relationship between MSSC and the overall System Safety Case must be made clear and consistent in both the System Safety Case and the Modular Software Safety Case.

In support of the treatment of failures by the overall Safety Case each individual block related software SC module may provide assurance that:-

a) The software block upholds its guaranteed behaviour and properties in the absence of a failure.

b) All credible failure mechanisms and circumstances under which the software block may fail to uphold its guaranteed behaviour or properties are identified.

Any failure mechanisms and circumstances under which the software block may fail to uphold its guaranteed behaviour, or properties which are not addressed within the software block will need to be addressed outside the software block, or shown to be of low enough risk to be acceptable for the given context of operation.

The analysis of the dependency-guarantee contracts between and dependency-guarantee relationships within Blocks allows dependencies which propagate through the software system to be followed, to ensure that the required behaviours are identified for all Blocks in the system. This means that MSSC can address explicitly the propagation of failures between modules.

9.3 System Integration Issues

Much of the Safety Case may be developed in independent SC Modules arguing about different parts of the system, but some post integration verification or system level analysis is expected to be necessary;

a) to provide evidence for claims which take no support from sub-system SC Module claims.

b) to reinforce SC module claims in the context of the overall system (see section 9.3.1).

c) for the System Level Hazard Analysis to consider all the safety relevant behaviour and properties of the sub-systems being integrated (see section 9.3.2).

9.3.1 SC Modules in the System Context

There are two complementary approaches to reinforcing the SC module claims in the context of the system.

1) The SC Module should satisfactorily capture the context in which its claims are valid, and this should be shown to be compatible with the system context.

2) After integration some further analysis or testing of the system is performed to supplement the SC Module claims.

9.3.2 Sufficiency of Hazard Analysis

When a SC Module is integrated into a system all its behaviour and properties should be provided to the Hazard Analysis.

A sub-system level analysis should have identified all behaviours and properties that might be hazardous, and this should be contextual information whenever the SC Module claims are used. The system level Hazard Analysis is context for the top level safety case.

During integration these two crucial pieces of information need to be verified as compatible. The Hazard Analysis may need to be revisited to achieve this.

9.3.3 Completeness and Compatibility

In support of the issues described in the previous two sections, SC integration activities include verifying **Safety Case Completeness** and establishing **Context Compatibility**.

Safety Case Completeness

A complete SC covers every aspect of the system relevant to its safety.

This includes a sufficient set of **Context** for the top level SC and each constituent SC module, such as the operating environment definitions or precise technical details of the claims.

Context Compatibility

The integrator must establish that the context in which the SC Module claims are made is not changed when those claims are integrated into the System SC.

Appendix A – Receptivity Analysis

Introduction

This evaluation worksheet provides an indication as to whether it is likely to be useful to produce a detailed engineering assessment and costing of the MSSC process to compare with alternative solutions.

This worksheet 'scores' the MSSC process across three benefit measures:

- Capability Agility
- Lifecycle Effort/Cost
- Programme Risk

The scoring is grouped under two categories of criteria:

- Programme – dealing with general aspects of conducting the development of a system.
- Technical – dealing with those aspects that directly impact the production of the software safety case,

This evaluation worksheet is intended to be printed, filled in and retained.

Instructions

Instructions for use of this evaluation worksheet are provided below, along with an example to illustrate how it should be used. Inputs required from the user are highlighted by grey text, where the grey text provides a prompt for how to fill in this field. For example;

The following instructions should be followed separately for both the Programme and Technical sets of questions to obtain a Suitability Indicator for each area.

1. Carefully read the question and select the response which best matches the programme under consideration.
2. Circle the score under each category for the selected response, and copy it into the Selected Score box underneath.
3. Fill in the Weighted Score by multiplying Selected Score by the Weighting.
4. Perform steps 1 to 3 for each question.
5. Sum the Weighted Scores for each category and record in the box provided (TWS).
6. Match the Total Weighted Scores for each category to the Suitability Indicator.

All questions must be answered.

Example

1.1 What type of development is this ?			
The type of development, whether a new system or an update to a legacy system, and the anticipated service life will impact how much benefit the MSSC process can provide.			
	Capability Agility	**Lifecycle Effort/Cost**	**Programme Risk**
New or replacement system with medium/long life	5	5	3
Legacy system update/revision with medium life	4	4	2
Short-life system (e.g. stop-gap)	3	1	2
Selected Score	S1.1A **4**	S1.1B **4**	S1.1C **2**
Weighting	30%	30%	30%
Weighted Score	WS1.1A **1.2** (S1.1A x Weight'g)	WS1.1B **1.2** (S1.1B x Weight'g)	WS1.1C **0.6** (S1.1C x Weight'g)

Assume this has been performed for all questions, and the sum of Weighted Scores is calculated and recorded as follows.

Total Weighted Scores	TWS 1A **2.8** (WS1.1A+WS1.2A +WS1.3A+WS1.4A)	TWS 1B **4.0** (WS1.1B+WS1.2B +WS1.3B+WS1.4B)	TWS 1C **2.2** (WS1.1C+WS1.2C +WS1.3C+WS1.4C)

Suitability Indicators - Programme			
	Capability Agility	**Lifecycle Effort/Cost**	**Programme Risk**
1.0 - 1.8	Strong –ve	Strong –ve	Strong –ve
1.8 - 2.6	Weak –ve	Weak –ve	Weak –ve
2.6 - 3.4	Neutral	Neutral	Neutral
3.4 - 4.2	Weak +ve	Weak +ve	Weak +ve
4.2 - 5.0	Strong +ve	Strong +ve	Strong +ve

Evaluation - Programme

1.1 What type of development is this ?			
The type of development, whether a new system or an update to a legacy system, and the anticipated service life will impact how much benefit the MSSC process can provide.			
	Capability Agility	**Lifecycle Effort/Cost**	**Programme Risk**
New or replacement system with medium/long life	5	5	3
Legacy system update/revision with medium life	4	4	2
Short-life system (e.g. stop-gap)	3	1	2
Selected Score	**S1.1A**	**S1.1B**	**S1.1C**
Weighting	30%	30%	30%
Weighted Score	**WS1.1A** (S1.1A x Weight'g)	**WS1.1B** (S1.1B x Weight'g)	**WS1.1C** (S1.1C x Weight'g)

1.2 Is the system expected to be adapted to meet emerging requirements ?			
The MSSC process provides greatest potential benefits to those systems that will undergo frequent minor adaptions throughout its life. A lack of, or infrequent major adaptions, are likely to incur little or no benefit from the MSSC process.			
	Capability Agility	**Lifecycle Effort/Cost**	**Programme Risk**
Never	3	1	2
1-3 major adaptions over life	4	3	3
More than 3 major adaptions over life	4	4	4
Frequent minor adaptions throughout life	5	5	5
Selected Score	**S1.2A**	**S1.2B**	**S1.2C**
Weighting	30%	30%	30%
Weighted Score	**WS1.2A** (S1.2A x Weight'g)	**WS1.2B** (S1.2B x Weight'g)	**WS1.2C** (S1.2C x Weight'g)

1.3 How many suppliers are involved ?			
Where multiple suppliers are involved, the MSSC process can help to manage the interfaces between the different organisations. The responses below are given in terms of small and large numbers; users should consider these terms relative to their prior experience of outsourcing development to multiple suppliers.			
	Capability Agility	**Lifecycle Effort/Cost**	**Programme Risk**
None – all development performed in-house	2	1	2
A small number of individual suppliers	3	3	3
A large number of individual suppliers	4	4	4
Selected Score	**S1.3A**	**S1.3B**	**S1.3C**
Weighting	20%	20%	20%
Weighted Score	**WS1.3A** (S1.3A x Weight'g)	**WS1.3B** (S1.3B x Weight'g)	**WS1.3C** (S1.3C x Weight'g)

1.4 Is there supplier experience with structured/modular safety cases ?			
Prior experience in developing structured safety cases using GSN, or modular safety cases (particularly use of the MSSC process), would be beneficial and make the programme more receptive to use of the MSSC process due to better isolation of change and increasing the authors likelihood of first time success in producing a compelling and structured modular argument. On the other hand, lack of such experience may limit the potential for realising the benefits due to increased effort in the familiarisation and training in use of the process.			
	Capability Agility	**Lifecycle Effort/Cost**	**Programme Risk**
None	2	1	2
Some use of GSN and structured safety cases	3	3	3
Some experience of developing Modular safety cases	4	4	4
Experience of applying MSSC process	5	5	5
Selected Score	**S1.4A**	**S1.4B**	**S1.4C**
Weighting	20%	20%	20%
Weighted Score	**WS1.4A** (S1.4A x Weight'g)	**WS1.4B** (S1.4B x Weight'g)	**WS1.4C** (S1.4C x Weight'g)

Total Weighted Scores	**TWS 1A (WS1.1A + WS1.2A +WS1.3A+WS1.4A)**	**TWS 1B (WS1.1B + WS1.2B +WS1.3B+WS1.4B)**	**TWS 1C (WS1.1C + WS1.2C +WS1.3C+WS1.4C)**

Suitability Indicators - Programme			
	Capability Agility	**Lifecycle Effort/Cost**	**Programme Risk**
1.0 - 1.8	Strong –ve	Strong –ve	Strong –ve
1.8 - 2.6	Weak –ve	Weak –ve	Weak –ve
2.6 - 3.4	Neutral	Neutral	Neutral
3.4 - 4.2	Weak +ve	Weak +ve	Weak +ve
4.2 - 5.0	Strong +ve	Strong +ve	Strong +ve

Evaluation - Technical

2.1 What is the size and/or complexity of the system ?			
The MSSC process provides greater potential benefits on large/complex systems and is not likely to worth the investment on small simple systems.			
	Capability Agility	**Lifecycle Effort/Cost**	**Programme Risk**
Small – typically a small number of items on a single processor	3	1	2
Medium – typically multiple items on multiple processors	4	3	3
Large – typically a large number of items on collections of processors	5	5	4
Selected Score	**S2.1A**	**S2.1B**	**S2.1C**
Weighting	20%	20%	20%
Weighted Score	**WS2.1A** (S2.1A x Weight'g)	**WS2.1B** (S2.1B x Weight'g)	**WS2.1C** (S2.1C x Weight'g)

2.2 What existing safety case is there ?			
An existing safety case which is aligned to system components may be easily modularised and would therefore make the legacy product more amenable to those change scenarios that affect it, whereas a poorly structured or monolithic safety case may be less receptive due to the extra work involved in modularising it. Where no safety case exists, the author has free reign over its modularity, also making that programme more receptive.			
	Capability Agility	**Lifecycle Effort/Cost**	**Programme Risk**
None	5	3	2
A legacy monolithic safety case (with no strong correspondence between claims and evidence and system components)	4	1	2
A structured (recent) safety case	3	2	3
A structured (recent) safety case with safety claims and evidence aligned to system components	3	4	3
Selected Score	**S2.2A**	**S2.2B**	**S2.2C**
Weighting	15%	15%	15%
Weighted Score	**WS2.2A** (S2.2A x Weight'g)	**WS2.2B** (S2.2B x Weight'g)	**WS2.2C** (S2.2C x Weight'g)

2.3 What is the variation of safety level(s) of the system ?			
Use of the MSSC can be beneficial when applied to systems with mixed safety levels, or where there is significant investment in establishing system safety, by separating areas of different safety involvement. At this time the MSSC process has been demonstrated for systems of low or medium safety levels so there may be additional risk associated with application to systems with higher safety levels.			
	Capability Agility	**Lifecycle Effort/Cost**	**Programme Risk**
Entirely low or no safety involvement	3	2	3
Predominantly low with some medium level safety involvement	5	5	5
Predominantly medium level safety involvement	5	5	5
Predominantly medium with some high level safety involvement	4	4	3
Predominantly high level safety involvement	4	3	1
Selected Score	**S2.3A**	**S2.3B**	**S2.3C**
Weighting	25%	25%	25%
Weighted Score	**WS2.3A** **(S2.3A x Weight'g)**	**WS2.3B** **(S2.3B x Weight'g)**	**WS2.3C** **(S2.3C x Weight'g)**

2.4 What independence of software modules will be supported by evidence ?			
If evidence will be provided to support an argument that the computing architecture provides strong software partitioning then a modular argument is going to be easier and more robust – so such a system is more receptive to MSSC.			
	Capability Agility	**Lifecycle Effort/Cost**	**Programme Risk**
No hardware or software partitioning	2	1	2
No hardware partitioning but software separation	2	2	2
Hardware supported partitioning	5	4	3
Hardware supported partitioning with a layered software architecture	5	5	5
Federated hardware system	3	2	3
Selected Score	**S2.4A**	**S2.4B**	**S2.4C**
Weighting	15%	15%	15%
Weighted Score	**WS2.4A** **(S2.4A x Weight'g)**	**WS2.4B** **(S2.4B x Weight'g)**	**WS2.4C** **(S2.4C x Weight'g)**

2.5 Will pre-existing software be safety-involved ?			
Where pre-existing software is being re-used its involvement with system safety and its available safety case could make a significant impact on the practicality of formulating a credible modular safety case.			
	Capability Agility	**Lifecycle Effort/Cost**	**Programme Risk**
None or very limited involvement (e.g. a tightly controlled interaction)	3	3	3
Some involvement for particular safety related functions, and with some supporting safety case for its use	4	4	4
General (widespread) involvement of software of unknown pedigree	1	1	1
General (widespread) involvement of software with an established safety-case	4	2	3
General (widespread) involvement of software with accompanying MSSC safety case	5	5	5
Selected Score	S2.5A	S2.5B	S2.5C
Weighting	25%	25%	25%
Weighted Score	WS2.5A (S2.5A x Weight'g)	WS2.5B (S2.5B x Weight'g)	WS2.5C (S2.5C x Weight'g)

Total Weighted Scores	TWS 2A (WS2.1A + WS2.2A +WS2.3A+WS2.4A+WS2.5A)	TWS 2B (WS2.1B + WS2.2B +WS2.3B+WS2.4B+WS2.5B)	TWS 2C (WS2.1C + WS2.2C +WS2.3C+WS2.4C+WS2.5C)

Suitability Indicators - Technical			
	Capability Agility	**Lifecycle Effort/Cost**	**Programme Risk**
1.0 - 1.8	Strong –ve	Strong –ve	Strong –ve
1.8 - 2.6	Weak –ve	Weak –ve	Weak –ve
2.6 - 3.4	Neutral	Neutral	Neutral
3.4 - 4.2	Weak +ve	Weak +ve	Weak +ve
4.2 - 5.0	Strong +ve	Strong +ve	Strong +ve