

CONTRACTING FOR ARCHITECTURAL, CLAIMS AND EVIDENCE ASSURANCE FOR MILITARY AVIATION SYSTEMS

Squadron Leader D.W. Reinhardt

Royal Australian Air Force
Deputy Senior Design Engineer
Avionics C-130H/J
RAAF Richmond, NSW, Australia

derek.reinhardt@defence.gov.au

Professor J.A. McDermid OBE FREng

Head of Department of Computer Science
University of York
United Kingdom

john.mcdermid@cs.york.ac.uk

Abstract

Contracts are instruments which provide a legally binding agreement for the purchase/exchange of goods or services. While both civilian and military aviation systems are acquired by contract, there are key differences in the role of contracts between the military circumstance and the civilian circumstance.

Civilian aviation system acquisition contracts are typically for the prospective owner of the aircraft to purchase an aircraft or modification that is separately subject to civilian National Airworthiness Authority certification under nationally legislated regulation. Therefore, in the civil context, the contract is usually not responsible for communicating, managing or enforcing of the activities of airworthiness certification; other than perhaps requiring that successful contract completion and payment be contingent on successful airworthiness certification with the relevant authority.

However, as many militaries are not just owners and operators but are also regulators/airworthiness authorities, then the use of contracts for the acquisition and modification of aviation systems means that the contract also becomes the proxy by which the regulatory functions are achieved. In simple terms, aircraft Type or system certification is managed via the contract, rather than via legal means such as nationally legislated regulation. This places additional requirements on contracts for military aviation systems.

In previous papers, the authors have proposed a framework for the assurance of software in aviation systems that involve an architecturally centric product approach to claims and evidence safety assurance. This framework differs from some other contemporary approaches in that it specifically requires that evidence and claims be established with respect to the suitability of specific product behaviours; thus any contract using this framework, must include enablers for these behaviours (and associated claims and evidence) to be communicated for the purposes of safety evaluation. The contract must also enable the findings of the safety evaluation to influence the product outcomes of the system at a phase of the lifecycle when this brings most benefit (usually earlier rather than later). This is somewhat different from the more prescriptive process oriented standards, which may be more concerned with process compliance and activity completion.

This paper examines the means by which the proposed assurance framework might be contracted for in the acquisition or modification of military aviation systems. Contracting paradigms including the preparation and execution of open tenders, restricted tenders, and sole source contracts are examined to determine how contracting for the proposed assurance framework might differ between these circumstances. Guidance is provided on contracting for architectural, claims and evidence assurance.

Keywords: Architecture, Assurance, Aviation Systems, Contracts, Fault Tolerance, Safety, Software Assurance, Software Safety, Tender.

1 Introduction

In complex aviation systems involving technologies (e.g. software) whose faults are dominated by a class of faults referred to as systematic faults, there are substantial challenges at providing assurance that these faults do not lead to unacceptable aircraft failure conditions. Because of these challenges, there have been limitations with traditional approaches to providing assurance of the suitability of the product behaviours of software in aviation systems (refer to [JTM07], [McD07], [McK06], [NTS06], and [Wea03]).

To provide an alternative to the traditional approaches that is more resilient with respect to the identified limitations, the authors have proposed a framework for the assurance of software in aviation systems. The framework involves an architecturally centric approach to coordination of claims and evidence with respect to safety assurance (refer to [ReM10] and [RMc10]). It differs from other contemporary approaches in that it specifically requires that evidence and claims be established with respect to the suitability of specific software behaviours with relationships to architecture and the fail safe design criteria. The framework is summarised in Section 2.4 of this paper.

An aspect of the suitability of the proposed framework that has not yet been considered is the extent to which it is compatible with the means by which regulatory outcomes and safety objectives are achieved in the military aviation circumstance. For reasons which shall be elaborated in Section 2.1 of this paper, the mechanism by which airworthiness regulatory outcomes are achieved for many military aviation systems during initial purchase or for modifications is via the contract rather than via legislation. This means that it is vital that any proposed assurance framework be compatible with the contracting paradigms used in military acquisitions, but without undermining the effectiveness of the assurance framework. In this context, there are numerous factors which might constitute success from both perspectives. In the contracting space, contracts which provide cost and schedule certainty are preferred by both suppliers and acquirers. Of course suppliers will also have a vested interest in profitability and acquirers in value for money. In the assurance space, acquirer's regulatory authorities will strive for achievement of an acceptable level of safety without significant or out of scope rework to treat risks, or without the retention of risks.

This paper examines the means by which the proposed assurance framework might be contracted for in the acquisition or modification of military aviation systems. Contracting paradigms including the preparation and execution of open tenders, restricted tenders, sole source contracts will be examined.

2 Background

Contracts are instruments which provide a legally binding agreement for the purchase/exchange of goods or services. A contract normally consists of terms and conditions, and is supported by technical annexes to define the goods/services and scope of work. For aviation systems, contracts are used for the acquisition and/or modification of these systems between the developer/manufacturer (i.e. supplier) and the owner or operator (i.e. acquirer). While both civilian and military aviation systems are acquired by contract, there are key differences in the role of contracts between the military circumstance and the civilian circumstance. The following section examines the origin and effect of these differences with respect to contracts, and provides emphasis why specific evaluation in the context of the military contracting environment is necessary.

2.1 Civilian versus Military Certification Environments

2.1.1 Civilian Certification Environment

Firstly, let's consider the civil aviation certification environment. The Federal Aviation Administration (FAA), the civil aviation airworthiness regulator in the United States of America, issues certification for new and modified aircraft that include software. This largely

independent certification is relied upon by the customers (owners and operators) who purchase and operate the aircraft. The FAA regime is also largely common to other civil aviation National Airworthiness Authorities (NAAs) around the world (e.g. Australia – Civil Aviation Safety Authority CASA, UK – Civil Aviation Authority CAA, Europe – European Aviation Safety Agency EASA). In this environment the roles of the developer, manufacturer, owner, operator and regulator are typically separated amongst different organisations or entities. This separation affords each of these entities some opportunity for independence in their function. For example, the owner and operator might be the same organisation (e.g. Qantas), whereas the supplier/developer/manufacturer might be Airbus or Boeing, and the regulator is a government agency (e.g. Civil Aviation Safety Authority (CASA) or the Federal Aviation Administration (FAA)). In addition, the prime developer and manufacturer are supported by a suite of sub-contractors that develop and manufacture aircraft systems and subcomponents.

Important to note is that the regulator is supported by regulations that are indoctrinated in law, and are therefore legally enforceable by the regulator onto those to which they apply (developer and manufacturer). For example, in Australia the Air Navigation Act 1920 [ANA20], and the Civil Aviation Act 1988 [CAA88] that define 'Australian' aircraft for which Civil Aviation Safety Regulations (CASRs) are promulgated and enforced by CASA. A similar arrangement exists within the US environment (Code of Federal Regulations (CFR), Title 14 Aeronautics and Space, Volume 1-5 defines aviation regulations [14CFR]).

Civilian aviation system acquisition contracts are typically for the prospective owner of the aircraft (e.g. Qantas) to purchase an aircraft or modification that is separately subject to civilian aviation airworthiness authority certification under nationally legislated regulation that has been outlined above. For example, the aircraft may already be type certified by a National Airworthiness Authority such as the FAA or CASA, or the aviation system has been subject to supplemental type certification (by similar authorities) or equivalent certification for minor modifications. Therefore, in the civil contracting context, the contract is usually not responsible for communicating, managing or enforcing of the activities of airworthiness certification; other than perhaps requiring that successful contract completion and payment be contingent on successful airworthiness certification with the relevant authority. In simple terms, in the civil context, the contract is not usually the means by which airworthiness certification is achieved.

Note though that the use of subcontractors to develop systems for an civil aircraft prime manufacturer does differ from the circumstances outlined above and would require elements of the certification to be achieved by the contract, however, such contracts are still not required (or necessary) to communicate detailed requirements for conduct of regulatory functions by the acquirer. While responsibility to achieve certification rests with the developer, and is certainly in the acquirer's interest, the regulator functions are managed by regulated interfaces to a separate independent authority (e.g. the FAA, as regulated by [14CFR21]).

2.1.2 Military Aviation Certification Environment

There are however other certification regimes where the regulator is not so overtly independent of the acquirer. Unlike civilian aviation arrangements, many militaries around the world are owners, operators and regulators; and to some extent developers and manufacturers.

The militaries are their own regulators (or airworthiness authorities) because they require flexibility to do things civilian operators would never need, such as: low flying, combat, close proximity flying, special modifications, stores clearances, contingency maintenance, battle damage repair, and operational imperatives involving safety versus capability tradeoffs; none of which are regulated by the civil authorities. These military airworthiness authorities typically define regulations that govern the conduct of their activities, however unlike the civil regulations, these regulations are open to discretion by the military regulator/authority to

allow tradeoffs between providing capability and safety based on the current military climate (e.g. war operations, peace support, counter terrorism, humanitarian assistance, peacetime training, etc [AAP1000-D])

For example, in Australia the Air Navigation Act of 1920 [ANA20] defines ‘State’ aircraft and designates the Chief of the Air Force (CAF) as the Australian Defence Force (ADF) Airworthiness Authority for Air Force, Army and Navy aircraft. Through internal Defence Instructions (DI(G) OPS 2-2 ADF Airworthiness Management [OPS2-2]), airworthiness management is separated into technical and operational responsibilities. The instruction also distinguishes the functions of the airworthiness regulators (i.e. the entities that write the technical and operational regulations) versus the airworthiness authorities (i.e. the entities that are responsible for interpreting the regulations and making the discretionary tradeoffs (via risk treatment or retention) between capability and safety).

Unlike the civil airworthiness regulations, the military airworthiness regulations are typically described in military publications which constitute lawful orders to those military and civilian government staff applying them. However, they are not necessarily legally binding to those developers and manufactures supplying equipment to the military. Table 1 identifies the military airworthiness regulations applicable to airworthiness certification in the ADF, UK MoD and USAF.

Australian Defence Force	United Kingdom Ministry of Defence	United States Air Force
DI(G) OPS 2-2 ADF Airworthiness Management AAP7001.048 – ADF Airworthiness Manual AAP7001.053 – Technical Airworthiness Management Manual AAP7001.054 – Airworthiness Design Requirements Manual AAP8000.010 – ADF Operational Airworthiness Manual	UK JSP 318B (Regulation of the Airworthiness of MoD Aircraft)* UK JSP 553 (Military Airworthiness Regulations)* DEF STAN 00-970 Design and Airworthiness Requirements for Service Aircraft	US Air Force Policy Directive 62-6 USAF Airworthiness MIL-HDBK-516B Airworthiness Certification Criteria
<i>*Note: The UK has established a Military Airworthiness Authority (MAA), and is undertaking re-development of airworthiness regulations.</i>		

Table 1: Military Airworthiness Regulations

Therefore, it is apparent that there are differences between the role of the civil airworthiness authorities, and some military regulators and airworthiness authorities. The differences are particularly notable with respect to the level of independence of the regulator from the other entities in the certification environment (owner, and operator), and the potential for legal enforcement of their requirements. Outside the scope of airworthiness, separate provisions apply to safety in society, as well as the regulation of technologies that are not aircraft. However, similar parallels can be drawn with regards to the regulation of these other technologies (e.g. Ships, Vehicles, Weapons, etc), as well as other safety-related regulations (e.g. Occupational/Workplace Health and Safety Provisions) that have been identified in the aviation case.

Because many militaries are not just owners and operators, but are also regulators (in the capacity as an airworthiness authority), then the use of contracts for the acquisition and modification of aviation systems means that the contract also becomes the proxy by which the systems certification elements of regulatory functions are achieved. In simple terms, type or system airworthiness certification is managed via the contract, rather than via other legal means. This places additional requirements on contracts for military aviation systems.

2.1.3 Acquisition, Contracts, Certification and the Provision of Assurance Evidence

In the civil aviation environment, when an applicant (usually the developer and manufacturer of an aircraft) applies for Type certification (or minor or major changes to the Type design), the regulations require that the applicant provide the civil authority with sufficient evidence to enable the regulator to conduct certification. The regulator does not have to contract for or purchase the evidence required to conduct certification, as it is wholly within the applicant's interest to achieve certification. While it is useful from a cost and schedule perspective for the applicant to wholly understand the regulator's evidence requirements prior to commencing a program, the regulator has means to obtain additional evidence if required to demonstrate the safety of the system at any time throughout the program. As a last resort the regulator may even with-hold certification of the system in question.

However in the military system, aircraft and other defence equipment is typically acquired through contract by the same overall entity that is the regulator (e.g. the Australian Defence Force (ADF), the Commonwealth – owner, operator and regulator). While within the ADF, different organisations are responsible for the acquisition (e.g. Defence Material Organisation), and the regulation functions in support of certification (e.g. Airworthiness Coordination and Policy Agency, and Directorate General Technical Airworthiness) these different organisations are considered by the contractor, and the law, as a single entity – the Commonwealth. Hence, the contract is not only responsible for defining the functional or capability requirements, it is also responsible for leveraging airworthiness and safety requirements onto the contractor. The contract is the top level instrument that must require the production and the purchase of/access to the evidence necessary to achieve certification. Where any ambiguity exists, or there is a shortfall in evidence, it is open to contractual negotiation or dispute, rather than being bound by legal obligations of contractors to the military regulations.

Thus the key conclusion reached from examining the differences in civil and military contract circumstances is the requirement for the military contract to facilitate the provision of evidence supporting the assurance framework, and management of shortfalls or limitations in the provision of the evidence.

2.2 Contracting Paradigms

Many different contracting paradigms exist including fixed price, cost plus, time and materials, performance or outcome based frameworks, cost and schedule risk sharing arrangements such as accords or alliances, etc. Contracts also exist that incorporate elements of several of these paradigms into one single contract. Within these paradigms, contracts may be offered via open tender, restricted tender or sole source, depending on the customers needs/wants to limit the market place for a solution in exchange for some other perceived benefit (usually time, as a suitable solution from an existing supplier with respect to the capability requirements may already be available).

When it comes to the acquisition of largely non-developmental aircraft or aircraft systems, the Australian Defence Material Organisation prefers fixed price contracts. Fixed price contracts facilitate straightforward budgeting arrangements through their cost certainty, and transfer some cost and schedule management responsibility to the contractor. Highly developmental systems are less cost effective to contract for under fixed priced arrangements, as the unknowns underpinning the developmental aspects usually translate into significant cost and schedule risk margins costed in by the contractor, if it is possible to cost at all. In these cases cost plus, risk sharing arrangements such as accords or alliances, or just a really well managed time and materials contract can be more cost effective in the long run. However, there are still numerous examples of developmental systems being acquired under fixed price arrangements for the ADF. Irrespective of the contract type, the better the contractor's understanding of the provision of evidence requirements before

contract signature, and the better their understanding of how shortfalls in evidence are to be resolved within the contract, the better the likelihood of a favourable contractual outcome. A favourable contractual outcome both the supplier and acquirer is generally a pre-requisite for a favourable capability outcome also. For the purposes of relevance to the Australian Defence Acquisition environment and the types of contracts preferred, this paper focuses on fixed price contract arrangements.

2.3 Contracting for Assurance of Evidence

A value for money and on-time/on-budget fixed price contract will only be possible when both the acquirer's and contractor's views on the product and evidence requirements are closely aligned prior to contract signature. While this proposition also underlies many other contracting paradigms, and there will be potential read across to these frameworks, these additional frameworks will not be considered directly.

Evidence requirements in contracts are typically articulated through the Statement of Requirement (SOR), Statement of Work (SOW) and Contract Data Requirements List (CDRL) which references applicable Data Item Descriptors (DIDs) for each piece of evidence the acquirer (in the capacity as an owner, operator and regulator) requires to conduct acceptance and certification. If an element of evidence or relevant design standard is not articulated through the contract, then it is unlikely that the developer or manufacturer will make this evidence available. In some cases the developer may not even produce the evidence on the understanding that it is not required by the acquirer. In the event that there are evidence shortfalls, and the contract does not cater for this circumstance, the acquirer will usually be required to seek an amendment to the contract (e.g. through a Contract Change Proposal (CCP)), which will usually incur a significant associated cost (and schedule impost). In addition, although DIDs are referenced which define the content requirements for evidence (i.e. DIDs are generally structural), DIDs do not necessarily define the quality of the information that underlies the required content (e.g. forms of argument, defensibility of the argument, quality of evidence, etc.). An acquirer review and acceptance cycle is usually the means of assuring the quality of the content of artefacts delivered against DIDs, but are usually constrained by the activities described in approved plans from earlier in the system or software engineering lifecycle. On this basis, it is possible to infer that contract DIDs need to be accompanied by material on the quality of argument and evidence, and compliance with the framework.

Ultimately, the acquirer needs a means to assure that the provision of sufficient ("enough") evidence is articulated through the contract, prior to contract signature, to ensure successful certification. Where there are shortfalls, these would ideally be minor in nature, and resolvable without addition or amendment to the contract. Furthermore, it is in the acquirer's interest to ensure that the contractor fully understands the evidence requirements throughout the tender/proposal process, to assure the accuracy of bids, and through contract negotiations where this understanding is refined and the final cost and schedule agreed. Similarly, the contractor requires a clear understanding of the evidence requirements in formulating their associated cost and schedule estimates.

Under the prescriptive software standards, this was relatively straightforward, as a software assurance standard and relevant SIL or DAL could be specifically called out in the contract, if required; or at least an agreeable framework for determining SIL and DAL assignment could be defined. The processes typically assured the acquirer of the provision (or at least the generation) of a certain known body of software evidence (analysis and test) from the activities prescribed in the standards. Recognising numerous limitations of current software assurance standards, some frameworks also contract for the provision of additional plans (e.g. software safety program plan) to provide for further insight into the contractors software activities. It is preferred that such plans are provided as part of tender responses, to allow maximum insight into whether the contractor is capable of meeting the required software safety and assurance objectives. As these plans typically only mature subsequent to contract

signature, there is often limited leverage for the acquirer to substantially vary any of these additional activities if they are subsequently assessed as insufficient.

However, under approaches such as those suggested by [Kel98], [Wea03] and [CAP670]; the concept of SILs or DALs is removed in favour of the flexibility of a contextually dependent safety argument, including the software system safety argument. While these approaches tie evidence to arguments, and [Wea03] and other subsequent work has made some steps to defining assurance requirements for safety arguments, these approaches predominantly rely on consultative arrangements to ensure the provision and quality of argument and evidence throughout the development lifecycle. Such an approach thus tends to be incompatible in its current maturity with fixed price contracting arrangements which force the consultative elements of the work to pre-contract signature, rather than throughout the whole contract. While it is possible for the acquirer (regulator) to articulate how well assured the software safety argument should be (e.g. valid/near valid, high, medium, low as defined in [Wea03]), or that complementary forms of evidence are required (e.g. Analysis and Test requirements from [CAP670]), neither of these frameworks associated specific evidence sufficiency requirements with the assurance of the argument in such a way it could be compatible with a fixed price contract.

In response to the limitations identified with approaches such as [Kel98], [Wea03], [CAP670] and other argument based approaches, the authors of this paper developed an alternative framework that sought to merge the product argument approach with some of the benefits of prescription evidence in the traditional software assurance and safety standards. The framework utilises Architectural [ReM10], Claims and Evidence [RMc10] assurance levels to provide a product focused suite of assurance criteria for aviation systems. In doing so, it was a goal to define a framework that could potentially be subject to contracting under fixed price contracting arrangements. To do this, the framework and associated contract mechanisms must include enablers for the product behaviours of the system and software (and associated claims and evidence) to be communicated for the purposes of safety evaluation. The contract must also enable the findings of the safety evaluation to influence the product outcomes and resolve shortfalls, but such that these would be unlikely and minor in nature (and minimal cost/schedule impact) if encountered post contract signature. The framework should also enable an appropriate amount of architectural definition and assurance evaluation pre-contract signature, to support refinement of the contract to resolve deficiencies so far as is reasonably practicable.

2.4 Assurance Framework

The following sub-sections summarise the key elements of the framework described by [ReM10] and [RMc10]. Sections 3 through 8 of this paper then examine how the framework might be contracted to address the factors indicated in Section 2.2 and 2.3.

2.4.1 Architectural Assurance

Through the examination of several actual aviation systems, and the 25.1309 / AC 25.1309 Fail Safe Design Criteria, [ReM10] proposed that architectural assurances be commensurate with the degree of fault tolerance against systematic faults. This implied that at the top level of the frame work, assurance (Architectural Safety Assurance Level (ASAL)) is based on the architectural relationships between systematic faults and failures, and the aircraft level failure conditions (and associated severity) as shown in Table 2.

Failure Condition Severity ¹	Architectural Safety Assurance Level (ASAL)	Systematic Fault Tolerance
Catastrophic	ASAL3	At least three (3) diverse ² systematic faults are necessary for the aircraft failure condition to be realised
Hazardous / Major	ASAL2	At least two (2) diverse ² systematic faults are necessary for the aircraft failure condition to be realised
Minor	ASAL1	At least one (1) systematic fault is necessary for the aircraft failure condition to be realised
No Safety Effect	ASAL0	Systematic fault tolerance is not required, however the designer may choose to incorporate fault tolerance to provide assurance of system availability and reliability

1. The worst credible failure condition severity of loss of and malfunction of the aircraft function with which the system and its software is associated.

2. For a systematic fault to be diverse of another systematic fault, it must be shown that the activation of one fault does not automatically lead to the activation of another systematic fault. In practice this is achieved by ensuring that the faults must occur in independent components and/or at differing layers of abstraction (e.g. software, LRU, system) where the correct functioning of the subsequent detection and handling mechanisms can be shown to be independent of the initiating fault condition or the detectable class of fault at the next layer is distinct of the initiating class of fault.

Table 2: Architectural Safety Assurance Level

Recognising that architecture, and the fault tolerance features exhibited by the architecture, are fundamental to providing protection against systematic faults, the ASAL concept provides the following criteria for the extent to which architectural components should provide for the absence or detection/handling of systematic faults as per Table 3.

ASAL	1st Absence/Detection and Handling Mechanism	2nd Detection/Handling Mechanism	3rd Detection/Handling Mechanism
ASAL3	Software Level	Partitioned Software Level [#] or LRU Level [†]	LRU Level [*] or System Level
ASAL2	Software Level	Partitioned Software Level [#] or LRU Level or System Level	Not Required
ASAL1	Software Level OR Line Replaceable Unit (LRU) Level OR System Level	Not Required	Not Required

must be independent of the initiating failure and the 1st Absence / Detection and Handling mechanism (i.e. through a partitioning mechanism)

* must be independent of the proceeding detection/handling mechanism

Table 3: ASAL Architecturally Layered Fault Tolerance

2.4.2 Claims Assurance

Having established the presence or requirements for the introduction of layers of absence or detection/handling mechanisms per the architectural assurance criteria in Section 2.4.1, each of these mechanisms is now considered in generic form as a 'constraint' on the behaviour of the software or system. For each constraint, a Claims Safety Assurance Level (CSAL) is determined, through its relationship to the previously defined ASAL (refer to Table 4).

ASAL	1 st Absence/Detection and Handling Mechanism		2 nd Detection/Handling Mechanism		3 rd Detection/Handling Mechanism		Additional Detection/Handling Mechanisms	
							Potentially Interfere ¹	Can't Interfere ²
ASAL3	Software Level	CSAL3	Partitioned Software Level [#] or LRU Level [*]	CSAL3	LRU Level [*] or System Level	CSAL3	CSAL2 [§]	CSAL0
ASAL2	Software Level	CSAL2	Partitioned Software Level [#] or LRU Level or System Level	CSAL2	Not Required		CSAL2 [§]	CSAL0
ASAL1	Software Level OR LRU Level OR System Level	CSAL1	Not Required				CSAL1	CSAL0

1 Potentially interfere with subsequent detection and handling
2 Can't Interfere with subsequent detection and handling
must be independent of the initiating failure and the 1st Absence / Detection and Handling mechanism (i.e. through a partitioning mechanism
* must be independent of the preceding detection/handling mechanism
§ additional mechanisms behaviour must be assured to reason that it won't interfere with the main mechanisms

Table 4: ASAL to CSAL relationship

The CSAL is a measure of the degree to which uncertainty in the behaviour of the constraint might exist, and provides guiding principles for the provision of evidence with respect to the 'constraint' (refer to Table 5).

Claims Safety Assurance Level (CSAL)	Category	Definition	Guiding principles for the substantiation of claims and provision of evidence with respect to satisfaction of attributes of software lifecycle products
CSAL 4 (not used)	Absolute Assurance	<u>Intended</u> and <u>unintended</u> behaviours of the absence or detection and handling constraint are absolutely assured with respect to safety, such that there is <u>no uncertainty</u> in behaviour	Not practicable (or affordable) to make this argument compelling – Near Absolute Assurance provides sufficient control of the uncertainty
CSAL 3	Near Absolute Assurance	All reasonably practical and effective steps have been taken to <u>systematically account</u> for the <u>intended</u> and <u>unintended</u> behaviours of the absence or detection and handling constraint with respect to safety, such that the remaining uncertainty would unlikely lead to a violation of the constraint under any circumstances	<ul style="list-style-type: none"> Specified behaviours with respect to the constraint Refined behaviours with respect to the constraint Implementation behaviours with respect to the constraint Introduced or generated behaviours (e.g. from translation or code generation toolsets) that may violate the constraint Target Computer behaviours that may violate the constraint Conditions or behaviours external to the constraint, but internal to the system, that may violate the constraint Conditions or behaviours external to the system that may violate the constraint

CSAL 2	Nominal Assurance	Steps have been taken to <u>systematically account</u> for the <u>intended functional behaviours</u> of the absence or detection and handling constraint with respect to safety, such that the remaining uncertainty would only lead to a violation of the constraint under unexpected circumstances	<ul style="list-style-type: none"> Specified behaviours with respect to the constraint Refined behaviours with respect to constraint Implementation behaviours with respect to constraint Target Computer behaviours with respect to the constraint Conditions or behaviours external to the constraint, but internal to the system, that may violate the constraint
CSAL 1	Limited Assurance	Claims broadly account for the <u>intended functional behaviours</u> of the absence or detection and handling constraint with respect to safety, such that the remaining uncertainty could lead to a violation of the constraint, but this would not be expected under normal operating conditions that would exercise the constraint	<ul style="list-style-type: none"> Specified behaviours with respect to the constraint Implementation behaviours with respect to the constraint
CSAL 0	No Assurance	No evidence exists to assure the absence or detection and handling constraint with respect to safety	No evidence

Table 5: Claims Safety Assurance Level

A set of attributes is defined that underpins the CSAL concept (refer to Annex A). Attributes are defined with respect to each of the key lifecycle products associated with software including:

- specified constraint level requirements (related to the absence or detection/handling mechanism associated with the ASAL),
- refined abstract level requirements (where relevant to design refinement/decomposition),
- low level requirements (requirements from which no further information or decomposition is required to develop source code for the nominated target computer),
- software source code, and
- executable object code.

Table 6 shows an extract of attributes from the specified constraint level requirements set. The table also details the impact of the attribute not being satisfied, and specifies the relationship to the CSAL. The relationship to the CSAL is expressed in terms of ‘tolerability of limitations’ which is elaborated in the subsequent section on evidence assurance.

Attribute	Impact of NOT Satisfying	CSAL3	CSAL2	CSAL1	CSAL0
Developed and Defined	Specified Constraint Level Requirements for constraint {constraint} do not exist – therefore there is no basis for the relevant behaviour existing in the software	Intolerable	Intolerable	Intolerable	Tolerable
Valid – Accurate / Consistent / Complete	Specified Constraint Level Requirements for constraint {constraint} exists but the specification of the constraint is invalid – therefore, there is potential for other lifecycle products or translations to refine or implement the behaviour erroneously	Intolerable	Constrained	Constrained	Tolerable

Table 6: Example of CSAL attributes for Specified Constraint Level (refer to Annex A)

2.4.3 Evidence Assurance

Having specified the tolerability of limitations in evidence per Annex A, the sufficiency of evidence (Evidence Safety Assurance Level (ESAL)) based on relevance, trustworthiness and the results of the evidence can be determined using Table 7. This provides a measure of the sufficiency of evidence from methods and techniques applied with respect to each attribute (from Table 6).

Tolerability of Limitations to Assuring Attribute	Relevance of Evidence	Trustworthiness of Evidence	Results of Evidence
Intolerable (ESAL3) – <i>limitations in evidence would be intolerable</i>	<p><u>No</u> limitations to the collective relevance of the method or methods' with respect to the attribute</p> <p>Limitations of each method or technique are systematically identified and treated by the application of complementary methods and techniques.</p>	<p><u>No</u> limitations to the evidence's trustworthiness with respect to the attribute.</p> <p>Limitations of the trustworthiness of evidence are systematically identified and treated by the application of appropriate competencies, reviews and inspections, and independence.</p>	<p>The results of the method or methods provides evidence of satisfying the attribute AND there is <u>no</u> counter evidence or potential source (uncertainty) of counter evidence to satisfying the attribute</p>
Constrained (ESAL2) – <i>limitations in evidence would be tolerable provided those limitations are constrained with respect to relevance, trustworthiness and results</i>	<p><u>Constrained</u> limitations to the method/s relevance with respect to the attribute</p> <p>Limitations of each method or technique are systematically identified and treated where practicable by the application of complementary methods and techniques. Non-treatment of a limitation should not introduce uncertainty grossly disproportionate to the limitation such that it would likely lead to a violation of the constraint</p>	<p><u>Constrained</u> limitations to the evidence's trustworthiness with respect to the attribute.</p> <p>Limitations of the trustworthiness of evidence are systematically identified and where practicable treated by the application of appropriate competencies, reviews and inspections, and independence.</p> <p>Non-treatment of a limitation should not introduce uncertainty grossly disproportionate to the limitation such that it would like lead to a violation of the constraint</p>	<p>Results of the method or methods provides evidence of satisfying the attribute AND counter evidence to satisfying the attribute is <u>limited</u> such that it would not likely lead to violation of the constraint</p> <p>Uncertainty is constrained such that counter evidence is unlikely.</p>
Tolerable (ESAL1) – <i>limitations in evidence would be tolerable</i>	<p><u>Notable</u> limitations to the method or method's relevance with respect to the attribute.</p> <p>Limitations of each method or technique may not be systematically identified and treated where practicable by the application of complementary methods and techniques.</p>	<p><u>Notable</u> limitations to the evidence's trustworthiness with respect to the attribute.</p>	<p>Results of the method or methods may provide evidence of non-satisfaction of the attribute and/or violation of the constraint OR counter evidence indicates possible violation of the constraint OR uncertainty may be substantial</p>

Table 7: ESAL Definitions

Ultimately, for each 'constraint' provided to establish architectural fault tolerance, evidence and argument should be provided at the specified CSAL for each 'attribute' to demonstrate the 'tolerability of limitations' with respect to the 'relevance', 'trustworthiness' and 'results' of evidence produced from the methods and techniques used by the developer. The case for tolerability is then subject to evaluation by an independent (from development) representative

on behalf of the certification authority and the impact of the top level product assurances evaluated from the tolerability of limitations with respect to each constraint.

3 Acquisition Paradigms

As introduced earlier, the three most common acquisition paradigms are the open tender, restricted tender or sole source acquisition. The three approaches offer a different means to getting on contract with a chosen supplier, depending on the following factors:

- the extent to which the solution will be developmental or off-the-shelf;
- the extent to which a supplier or suppliers are known prior to the acquisition;
- the extent to which engaging a larger market improves competition and value for money in solutions offered; and
- the extent to which engaging a narrower market improves contractual response times and introduction to service.

Consideration of these factors amongst others permits the determination of the acquisition approach. The following sub-sections elaborate the three different approaches.

3.1 Open Tender

The Open Tender involves the release of a Request for Tender (RFT) to the whole market, a potentially large number of prospective tenderers. The RFT typically contains a version of the contract Statement of Work, and Statement of Requirement (usually referred to as a Tender SOW and Tender SOR respectively) for the purposes of tendering against. The RFT responses would then be evaluated and a preferred tenderer identified, with which contract negotiations would commence. At the time of contract negotiations a draft contract is prepared based on the original tender documents, and refined based on the preferred tenderers RFT response. Presuming the contract negotiations are successful, contract signature would be achieved.

Note that some tender processes involve an initial release of a Call for Expressions of Interest to identify the market, and then release of the RFT to only suitable responses to the EOI. This approach is really a hybrid of the open tender and restricted tender, but with the luxury that the actual tender SOW and SOR can be refined based on the initial look to the market under the Call for Expressions of Interest.

3.2 Restricted Tender

The Restricted Tender involves the release of the Request For Tender (RFT) to a restricted number of market participants. This subset of market participants will have been pre-determined either by a market selection activity (such as a Call for Expressions of Interest, Request for Proposals, etc), or through market research. For example, for very specialised technologies and systems, it may be possible for the acquirer to know that the only organisations capable of working with such technologies and systems, may be those that they've identified via market research. This may only be an assumption though and may benefit from periodic re-examination of the market, perhaps by the Open Tender process described above. The key factor that distinguishes this approach from the Open Tender is that the tender is restricted to a nominated number of two or more tenders. Otherwise the RFT release, tender evaluation, contract negotiations, contract signature and contract execution all follow very similar processes to the Open Tender.

3.3 Sole Source Acquisition

The Sole Source Acquisition involves confining the acquisition to a single supplier, usually because the supplier has been predetermined to provide an off-the-shelf solution, or because through prior determination, the supplier has been assessed as the most suitable entity to

provide the system within the circumstances of the acquisition. Examples of some circumstances include rapid acquisitions due to operational imperatives, and intellectual property restrictions that prevent the work being contracted to another supplier.

For the Sole Source Acquisition, the Request For Tender (RFT) is usually replaced by a Request For Quote (RFQ) or Request for Proposal (RFP) to reflect the more definite nature of the acquisition. In some cases the proposal request should be very similar in nature to an RFT, as all the same information needs to be communicated. However, the authors' experience is that sometimes this step is overlooked by the project authority, because of perceptions that the project scope is already defined by the solution. While this perception may be true for physical tangibles, it is less applicable to the body of evidence needed to form the safety case. Regardless of the cause, the outcome is that for the Sole Source Acquisition, the proposal evaluation and contract negotiations phase usually wears a much greater burden for resolving evidence requirements into the SOW and SOR than for the Open Tender and Restricted Tender paradigms. If it is overlooked then the contract will likely be inadequate, and result in certification challenges. Provided the evidence requirements are resolved pre-contract signature, then once on contract, there should be limited differences between the Sole Source approach and the Open and Restricted Tender approach.

4 Project Definition and Approval

Project Definition and Approval is the project lifecycle phase that precedes any Tender Preparation and RFT Release. The intent of this lifecycle phase is to define the acquisition requirements, either based on a capability requirement, or a known market solution that has already been established to fulfil a capability requirement. In addition to the functional/capability requirements for the acquisition, the project definition and approval phase also estimates the project costs and schedule. These additional factors are important, because they are key determiners in project approval by the project approval authority. For example, the Australian Government uses a two pass process for acquisition approvals, with the first pass providing some preliminary funding and time allocation to permit solutions (including costs and schedule) to be explored under RFT and RFQ processes prior to second pass approval. Second pass approval sets the project scope, costs and schedule, and provides approval to proceed to contract signature. The emphasis of the first pass phase is to increase the confidence in the project, cost and schedule estimates, such that the risk of proceeding into contract signature is constrained to a level deemed tolerable by the project approval authority (which is usually Government, for major capital Defence acquisitions).

So what does this process mean for Architectural, Evidence and Claims assurance approach? The project definition and approval phase is therefore the timeframe when the Rough Order of Magnitude (ROM) or notional costs for evidence to be provided in support of the safety case should be determined. These ROM costs are important to define, because they set expectations for the cost of achieving safety in the acquisition, which is in addition to more material acquisition costs (such as the delivery of equipment and/or services).

From the Architectural, Evidence and Claims assurance perspective, the project definition and approval phase is where the notional costs should be determined for:

- system architecture and software architecture to meet ASAL requirements – what level of redundancy, diversity and fault tolerance is likely to be required? And how much will this likely cost?
- quantity and type of constraints – how many constraints are required to provide the requisite determinism regarding the knowledge of behaviours of the system and its software under normal and failure circumstances? How critical are these constraints?
- likely breadth and depth of evidence to meet ESAL criteria for constraints at notional CSALs – how much time and money will be required for evidence generation, both proactively and retrospectively (when circumstances dictate)?

The intent of having relative certainty as to these factors is to ensure that notional funding can be established and sought from the funding authority prior to tender release. Understanding of these factors helps to set Government, acquirer and ultimately supplier expectations on the realistic costs of the project. When realistic costs are known, more realistic project scope and solutions can be established. In an environment where part of the sell to establish and get approved a project will be to emphasise the substantial capability it provides, often there will be down-playing the real cost and schedule implications. Some contractors also use contractual tactics that seek to downplay costs prior to contract signature, knowing there may be opportunities to recover costs and profit outside the initial contract (e.g. through Contract Change Proposals or separate sustainment contracts). It is the certainty during project definition and approval in these costs that can help combat such tactics.

To elaborate the usefulness of this information, let's consider several simple acquisition examples. Consider for example the acquisition of a fixed-wing aircraft that includes a simplex digital automatic flight control system providing stability augmentation with limited fault tolerance. Say for example also that, during certain phases of flight, a malfunction of the simplex digital AFCS leads to an erroneous deflection of control surfaces requiring exceptional pilot recovery. This scenario is credible failure scenario for a simplex system, and one that would normally require consideration under the fail safe design criteria. Knowledge of the importance of the system during those phases of flight should prompt pre-emptive examination of the suitability of the simplex AFCS architecture and its level of fault tolerance before contract signature. A second example might be a flight management system on the same aircraft type that interfaces to dual air data sensors, but for which there is no software, LRU or system level treatment of erroneous air data which is flagged valid. Again, knowledge of the suitability of this proposed architecture and constraints is useful in establishing if it is even possible that the design would meet safety objectives. Even if it would turn out that these architectures and constraints would be suitable following appropriate analysis, verification and validation; then at least the project risk costs could be accurately modelled in determining the viability of the acquisition. Both these examples imply a level of insight into the design early in the project lifecycle and sometimes even before suppliers are engaged. However, in envisaging a potential solution (even in an entirely capability requirement focussed acquisition), estimation of the suitable solution space provides useful means of establishing the suitability of solutions proposed under tender or proposal phases.

Project Definition Funding (PDF), a type of funding in the ADF for enabling pre-first pass definition of capability options provides one means for engaging prospective suppliers and establishing the suitability of their design solutions before official tender/proposal mechanisms are pursued. This funding might be used to develop specific insight into the architectural, claims and evidence assurance of perspective options, the shortfalls in this regard, and the viability of resolving these under more formal contractual instruments.

The imposition of requiring the knowledge specified above however is that the team responsible for project definition and approval submissions needs to have team members with skills sets that allow them to estimate the aforementioned factors. Alternatively, they should be able to access specialists that can provide the input. Capability and project definition authorities should examine the practicality of having architectural and assurance specialists at ready access. Their role should not be to constrain solutions at the expense of novelty or innovation, but to set expectations for the types of solutions that might be acceptable with respect to safety. However, as the skills in safety assurances will always be in short supply, project definition authorities would be aided by estimation tools and techniques to provide greater certainty and repeatability in these estimates for lesser skilled staff. The development of accurate estimation tools will only be possible once the more quantitative data on the costs of these architectural and assurance approaches is obtained.

This paper recommends that project authorities establish means of collecting this information on both legacy and new projects.

With the project scope defined and expectations established, the execution of the acquisition depends on the acquisition paradigm chosen. Sections 5, 6, and 7 detail the approaches for Open Tender, Restricted Tender and Sole Source Acquisition respectively.

Note that this paper primarily considers the contract relationship between the acquirer and the system integrator / prime software developer, and thus proposed Statement of Requirement (SOR) and Statement of Work (SOW) clauses have been established for that context. These clauses will require adaption for inclusion in contracts with lower tier contractors or vendor level contracts, although the intent of the clauses should remain similar.

5 Open Tender Acquisition

The following sub-sections outline the key elements of implementing the ASAL/CSAL/ESAL framework for each lifecycle phase of an open tender.

5.1 Tender Preparation

The tender preparation lifecycle phase is where the tender documentation is prepared. Tender documentation usually consists of a set of covering Tender explanation, draft terms and conditions (T&Cs), the Tender Statement of Requirement (SOR), and Tender Statement of Work (SOW).

One key trade off a tender must make is between the level of detail required from prospective tender responses and the impost this might place on prospective bidders in actually providing a tender response. As only a small percentage of tender responses are actually successful, and tenderers invest substantial resources in preparing them, the acquirer must be cognisant of avoiding deterring potential tenderers due to the level of effort required to tender. Therefore, in establishing the SOR and SOW clauses for the tender to implement the ASAL/CSAL/ESAL framework, the solution must provide for sufficient disclosure and understanding to achieve the objectives of the framework, but while ensuring the minimum impost of tenders. This is a difficult balance.

Where the acquisition or modification is of substantial complexity, then the single phase tendering process may not incentivise suppliers to invest a level of effort to develop their solution to a level that permits preliminary evaluation against the ASAL/CSAL/ESAL framework. In these cases a two-phase tender may be more suitable. The first phase would identify holistic solutions that accord with the safety objectives of the program and use a normal tender construct. The second would be a partially funded tender phase, where funding is provided to a restricted set of tenderers to further develop the tender artefacts supporting evaluation against the ASAL/CSAL/ESAL framework. The second phase would be more synonymous with a Restricted Tender, but include provision for funding.

5.1.1 Tender SOR Clauses

The Tender SOR nominally forms the document that lays down the capability, functional, safety, etc requirements for the system to be acquired. Its role is to set measurable product outcomes/requirements, and communicate them to the tenderer. The SOR usually consists of a set of both goal based and prescriptive requirements, depending on the acquirers requirements on flexibility/novelty in the design solution.

For the ASAL, CSAL and ESAL framework, the Tender SOR is where the top level 'product' hooks of the framework should be articulated. As the primary goal of the ASAL/CSAL/ESAL framework is safety assurance, the Tender SOR clauses for the framework should be located in the safety assurance section of the SOR.

The following are example SOR clauses to demonstrate how this might be achieved. The rationale for these clauses is provided in Section 5.2. Note the top level safety objective clauses have also been included to provide context to the ASAL/CSAL/ESAL framework clauses, and were adapted from clauses existing in the Australian Defence Force Contracting Templates [ASDEFCON] and [AAP7001.054].

Top-level Safety Goal

The [System Name] shall not cause a hazard to safety when operating in the intended roles, configurations and operating environments of the [Acquirer].

Criteria for Risk Treatment and Retention

The [System Name] shall meet the requirements of 14CFR25.1309¹, and all associated Advisory Circulars, Orders, and Notices.

The risk of the [System Name] causing a hazard to safety when operating in the intended roles, configurations and operating environments of the [Acquirer] shall be:

- *tolerable to the [Acquirer] per a risk management framework agreed by the [Acquirer]; and*
- *explicitly documented and communicated to the [Acquirer].*

Architectural Safety Requirements

The [System Name] design shall employ the fail safe design criteria of AC25.1309² to provide protection against both random and systematic classes of faults and failures, regardless of their origin.

The [System Name] architecture and mechanisms for achieving fault tolerance against systematic faults shall meet the Architectural Safety Assurance Level (ASAL) requirements defined in Table 2 of this paper.

The [System Name] shall meet the ASAL Architecturally Layered Fault Tolerance Requirements as defined in Table 3 of this paper; or be shown to provide an equivalent level of fault tolerance by alternative means.³

5.1.2 Tender SOW Clauses

The role of the Tender SOW is to define the contractor scope of work to produce evidence deliverable to the acquirer to show compliance with the aforementioned Tender SOR clauses. It sets the expectations for the level of evidence generation by the contract and delivery to the acquirer.

The following are example Tender SOR clauses to demonstrate how this might be achieved. Elaboration for the reasoning behind their inclusion is discussed in Section 5.2.

Informing Architectural Suitability

The [Tenderer] shall prepare a [Conceptual System and Software Architecture Suitability Document] to describe how the [System Name] architecture and mechanisms for achieving fault tolerance against systematic faults meets the Architectural Safety Assurance Level (ASAL) requirements defined in [Table 2 of this paper].

¹ [14CFR25] Subpart F – Equipment §25.1309 Equipment, systems, and installations sets the acceptable risk criteria for civil transport category airplanes.

² [AC25.1309] describes the acceptable means of compliance with 14CFR25.1309

³ An alternative means may be appropriate where the system architecture does not conform to the software, LRU and system level model used for expressing protection mechanisms against systematic faults in Table 3.

The [Tenderer] shall prepare a [Conceptual System and Software Architecture Suitability Document] to describe how each proposed constraint (i.e. absence/detection and handling mechanism) is proposed to achieve the ASAL Architecturally Layered Fault Tolerance Requirements as defined in [Table 3 of this paper]; or be shown to provide an equivalent level of fault tolerance by alternative means.

Assurance of Constraints using Claims Assurance (CSAL)

The [Tenderer] shall prepare a [Software Assurance Plan] to describe the Claims Safety Assurance Level (CSAL) proposed for each constraint described in the [Conceptual System and Software Architecture Suitability Document] as per [Table 4 of this paper].

Assurance of Evidence (ESAL and Tolerability of Limitations)

Defining the Evidence

The [Tenderer] shall prepare a [Software Development Plan] to describe the methods and techniques proposed to be used throughout the software development lifecycle, including description of techniques or methods used prior to this development but for which evidence is relevant.

The [Tenderer] shall prepare a [Software Development Plan] to describe how all evidence, both new and existing, or produced from the application of [Tenderer] proposed methods and techniques will be documented, stored, and retrievable.

The [Tenderer] shall prepare a [Software Development Plan] to describe how CDRLs [refer list below] will be produced per the schedule X.

Assessing the Evidence

The [Tenderer] shall prepare a [Software Assurance Plan] to describe how the evidence produced from the application of the [Tenderer] proposed methods and techniques is proposed to achieve the Evidence Safety Assurance Level (ESAL) requirements for tolerability of limitations as defined in [Table 7 to this paper]; for each attribute of each software lifecycle product [per Annex A to this paper], at the CSAL [defined per Table 5] and as described in the [Conceptual System and Software Architecture Suitability Document] for each proposed constraint.

The [Tenderer] shall prepare a [Software Assurance Plan] to describe the means, either via provision of evidence or via access provisions to tenderer facilities and data, for the [Acquirer] to inspect or review all evidence, both new and existing, from the application of [Tenderer] proposed methods and techniques for the purposes of certification evaluation by the [Acquirer].

Exemplar Elements of the Software Aspects of the Safety Case

The [Tenderer] shall prepare an [Exemplar Safety Case] to show the implementation of the ASAL, CSAL and ESAL framework for at least one constraint in each generalised category, type or class of constraint proposed. The [Tenderer] shall describe the set of categories, types or classes by which they have categorised the proposed constraints.

The secondary element of the Tender SOW is the Tender Data Requirement List (TDRL), and the associated Data Item Descriptors (DIDs). The following TDRL items relating to the aforementioned SOW clauses should be listed for delivery in the TDRL:

- Conceptual System and Software Architecture Suitability Document
- Software Development Plan
- Software Assurance Plan
- Exemplar Safety Case

Additional TDRLs may be listed per any additional acquirer's requirements.

Further, there will be additional CDRLs (and thus DIDs) that won't be deliverable under the Tender, but may be included in the Tender SOW to be costed as part of tender response but delivered or accessed under the contract. With respect to the implementation of this framework, these are as follows (refer to latter parts of the paper for information on their contents):

- *Safety Case* to summarise how the architectural fault tolerance contributes to mitigation of systematic failure modes
- *Software Configuration Index / Software Version Description* to summarise the configuration of software to which the evidence relates
- *Software Assurance Summary* to summarise the tolerability of limitations for evidence shortfalls with respect to software lifecycle product attributes CSAL and associated ESAL requirements.

The following additional DIDs (non-exhaustive list) may be listed in the CDRL depending on the acquirer's requirements to obtain intellectual property rights. These are optional with respect to this framework, and the acquirer's contracting policy and certification guidance documents should be sought for specific requirements regarding deliverables.

- System Development Specification
- Sub-System Design Document
- Software Requirements Specification
- Software Design Document
- Source Code Repository
- Toolset Repository
- Software Lifecycle Data Repository
- Software Verification Plan
- Software Verification Plan
- Software Verification Results
- System Verification Results

5.2 Tender Responses

The tender response is the phase of the tender process where the tenderer prepares their response documentation to the Tender SOR and SOW clauses defined above. While the onus of this activity is on each contractor, the section will elaborate the rationale for the inclusion of the aforementioned SOR and SOW clauses to assist tenderers in preparing their responses.

The Tender SOW requires the preparation of a Conceptual System and Software Architecture Suitability document to meet ASAL requirements including conceptual identification of absence and detection/handling mechanisms – i.e. constraints. The purpose of the tender requesting this information is to permit evaluation of the extent to which the holistic safety and software architecture requirements are costed into the tender response. The retrospective incorporation of constraints to treat systematic failure modes is rarely straightforward, particularly when it affects architecture. Therefore, it is in the acquirer's interests to establish the extent to which the contractor has determined an architecture based on the types of constraints required to meet safety objectives. While it is recognised that many sub-system architectures may not be well defined for large system acquisitions, the absence of this information in a tenderer's response will permit the acquirer to adjust the contractors proposed costing by a risk figure based on the amount of uncertainty (or extent of

suitability) in the tenderers proposed architecture to provide a normalised evaluation of tenderers responses that do include the relevant information. This paper hasn't defined the implementation of costing adjustments, and this is recommended as further work.

The Tender SOW also requires that, through the provision of the Software Development Plan, all evidence that has been or will be produced is documented, along with the methods and techniques used to product the evidence. This information has been distinguished from the Software Assurance Plan, as it already resembles existing Software Development Plan content requirements under existing frameworks.

More specifically for this framework the Tender SOW requires, through the provision of a Software Assurance Plan, the preliminary CSAL assignment for each constraint, along with the proposal of methods and techniques to satisfy ESAL criteria with respect to attributes at assigned CSALs (generic). The purpose of requesting this information to permit the certification authority to evaluate in the tender evaluation the breadth and depth of evidence proposed with respect to the general tolerability of limitations concept. It provides the acquirer a measure of the extent to which the proposed evidence is likely to meet with acquirer evidence benchmarks. It also permits contract authority determination during the tender evaluation as to whether the tender adequately costs the evidence, and what any evidence shortfalls might notionally cost.

The final element of the Tender SOR is the provision of exemplar attribute satisfaction for at least one constraint in each general class of constraint. The exemplar safety case population is intended to assist in establishing the contractor understands the application of the ASAL/CSAL/ESAL framework with respect to the objectives of the framework and the benchmarks for evidence that should be produced under the framework. It permits the certification authority to evaluate during the tender evaluation whether the exemplar attribute satisfaction is acceptable, and thus establish if the tenderer understands the tolerability of limitations in evidence provision.

5.3 Tender Evaluation

The tender evaluation is the phase of the tender lifecycle where the tender responses are evaluated against a set of pre-determined tender criteria that includes measures of:

- compliance with the Tender SOR, and SOW
- contractor costs estimates
- contractor schedule estimates
- adjusted cost and schedule estimates based on shortfalls or limitations in tenderer responses

With respect to the ASAL/CSAL/ESAL framework, the tender evaluation should include an evaluation of architectural suitability, and also the 'tolerability of limitations' to assuring attributes of proposal of methods and techniques to satisfy ESAL criteria with respect to attributes at conceptually assigned CSALs. It will also include an evaluation of 'tolerability of limitations' to assuring attributes of exemplar attribute satisfaction in the exemplar safety case. This part of the tender evaluation should be carried out of specialists with both system domain and safety and software assurance experience, all of who are familiar with the fault tolerance principles that underpin the ASAL/CSAL/ESAL framework.

The aim of the tender evaluation is establishment of the following:

- Certification Authority understanding of each tenderers' approach, including architecture, fault tolerance and provision of assurance evidence;
- the 'Tolerability of Limitations' in proposed evidence;

- the scope, as well as cost and schedule impact, of additional evidence required to resolve intolerable architecture or evidence deficiencies; and
- the final ranking of tenderers on the above factors.

On the basis of the established ranking, a preferred tenderer will be identified with which contract negotiations can commence.

5.4 Contract Preparation

The contract preparation lifecycle phase is where the contract documentation is prepared. Tender documentation usually consists of the covering contract explanation, terms and conditions (T&Cs), the Contract Statement of Requirement (SOR), and Contract Statement of Work (SOW). The following sections examine the clauses necessary to contract for the ASAL/CSAL/ESAL framework.

5.4.1 Contract SOR Clauses

The following are example SOR clauses, and are consistent with the SOR clauses issues at Tender phase (repeated for convenience only, and note the top level safety objective clauses have also been included to provide context to the ASAL/CSAL/ESAL framework clauses):

Top-level Safety Goal

The [System Name] shall not cause a hazard to safety when operating in the intended roles, configurations and operating environments of the [Acquirer].

Criteria for Risk Treatment and Retention

The [System Name] shall meet the requirements of 14CFR25.1309⁴, and all associated Advisory Circulars, Orders, and Notices.

The risk of the [System Name] causing a hazard to safety when operating in the intended roles, configurations and operating environments of the [Acquirer] shall be:

- *tolerable to the [Acquirer] per a risk management framework agreed by the [Acquirer]; and*
- *explicitly documented and communicated to the [Acquirer].*

Architectural Safety Requirements

The [System Name] design shall employ the fail safe design criteria of AC25.1309⁵ to provide protection against both random and systematic classes of faults and failures, regardless of their origin.

The [System Name] architecture and mechanisms for achieving fault tolerance against systematic faults shall meet the Architectural Safety Assurance Level (ASAL) requirements defined in Table 2 of this paper.

The [System Name] shall meet the ASAL Architecturally Layered Fault Tolerance Requirements as defined in Table 3 of this paper; or be shown to provide an equivalent level of fault tolerance by alternative means.⁶

⁴ [14CFR25] Subpart F – Equipment §25.1309 Equipment, systems, and installations sets the acceptable risk criteria for civil transport category airplanes.

⁵ [AC25.1309] describes the acceptable means of compliance with 14CFR25.1309

⁶ An alternative means may be appropriate where the system architecture does not conform to the software, LRU and system level model used for expressing protection mechanisms against systematic faults in Table 3.

5.4.2 Contract SOW Clauses

Unlike the Contract SOR clauses which have not been changed for the product implementation of the ASAL/CSAL/ESAL framework between tender and contract SORs, the contract SOW should now include clauses which reflect the acquisition of the system and its evidence, and not just simply the acquisition of the information to support a tender evaluation of the system. To achieve this, the following example SOW clauses are provided:

Architectural Assurance

The [Contractor] shall prepare a [System and Software Architectural Assurance Document] per CDRL XX to describe how the [System Name] architecture and mechanisms for achieving fault tolerance against systematic faults achieves the Architectural Safety Assurance Level (ASAL) requirements defined in [Table 2 of this paper].

The [Contractor] shall prepare a [System and Software Architectural Assurance Document] per CDRL XX to describe how each proposed constraint (i.e. absence/detection and handling mechanism) is proposed to meet the ASAL Architecturally Layered Fault Tolerance Requirements as defined in [Table 3 of this paper]; or be shown to provide an equivalent level of fault tolerance by alternative means.

Assurance of Constraints using Claims Assurance (CSAL)

Proposal of CSAL

The [Contractor] shall prepare a [Software Assurance Plan] per CDRL XX to describe the Claims Safety Assurance Level (CSAL) proposed at commencement of development for each constraint described in the [(Preliminary) System and Software Architecture Document] as per [Table 4 of this paper].

Achievement of CSAL

The [Contractor] shall prepare a [Software Assurance Summary] per CDRL XX to describe the Claims Safety Assurance Level (CSAL) established each constraint described in the [System and Software Architecture Document] as per [Table 4 of this paper].

Provision of Evidence (ESAL and Tolerability of Limitations)

Defining the Evidence

The [Contractor] shall prepare a [Software Development Plan] per CDRL XX to describe the methods and techniques proposed to be used throughout the software development lifecycle, including description of techniques or methods used prior to this development but for which evidence is relevant.

The [Contractor] shall prepare a [Software Development Plan] to describe how all evidence, both new and existing, or produced from the application of [Contractor] proposed methods and techniques will be documented, stored, and retrievable.

The [Contractor] shall prepare a [Software Development Plan] to describe how CDRLs XX-XX [refer list below] will be produced per the schedule [at Table 8].

Assessing the Evidence

The [Contractor] shall prepare a [Software Assurance Summary] to describe how the evidence produced from the application of the [Contractor] proposed methods and techniques achieves the Evidence Safety Assurance Level (ESAL) requirements for tolerability of limitations as defined in [Table 7 to this paper]; for each attribute of each software lifecycle product [per Annex A to this paper], at the CSAL [defined per Table 5] and as described in the [System and Software Architectural Assurance Document] for each constraint.

The [Contractor] shall prepare a [Software Configuration Index] to describe the configuration of the [System] and [Software] relevant to the evidence, claims and architecture described by the [Software Assurance Summary]

Software System Safety Case

The [Contractor] shall prepare a [Safety Case] per CDRL XX to describe how the safety objective, and safety assurance requirements of the contract SOR have been achieved for [System] and [Software], and to provide the argument and evidence to show the satisfaction of ASAL/CSAL/ESAL criteria for each constraint.

Certification Evaluation

The [Contractor] shall prepare a [Software Assurance Plan] to describe the means, either via provision of evidence or via access provisions to tenderer facilities and data, for the [Acquirer] to inspect or review all evidence, both new and existing, from the application of [Contractor] proposed methods and techniques for the purposes of certification evaluation by the [Acquirer].

The [Contractor] shall provide evidence or access to evidence as described in the [Acquirer] approved [Contractor]'s [Software Assurance Plan] for the purposes of certification evaluation by the [Acquirer].

Intolerable Limitations in Evidence, Claims or Architecture

Where the [Acquirer]'s certification evaluation establishes that the [Contractor] has not achieved the requirements of the ASAL/CSAL/ESAL framework, or there are shortfalls in the 'Tolerability of Limitations' of evidence then the [Contractor] shall undertake one or more of the following remediation actions to resolve the shortfalls to the satisfaction of the certification authority:

- engineering change to architectural constraints,
- engineering change to implementation of architectural constraints, or
- additional analysis, verification and validation by further or supplementary application of methods or techniques.

The [Contractor] shall amend all relevant deliverables per the CDRL to incorporate the engineering changes and additional evidence.

Note to Contractors

The above clause provides the means for the certification authority to address shortfalls against the ASAL/CSAL/ESAL framework. While this clause may be interpreted to result in unbounded programmatic risk for the contractor, the intent is to focus both acquirer and contractor efforts at establishing unambiguous consensus during the tender process and contract negotiations. The contractor should not sign the contract if they believe there remains substantial uncertainty regarding the provision of evidence against the ASAL/CSAL/ESAL framework, and instead request further clarification during contract negotiations.

The Contractor shall prepare the following deliverables and deliver them in accordance with the document delivery schedule defined in the CDRL:

- [System and Software Architectural Assurance Document] [CDRL XX]
- [Software Development Plan] [CDRL XX]
- [Software Assurance Plan] [CDRL XX]
- [Software Configuration Index] / [Software Version Description] [CDRL XX]
- [Safety Case] [CDRL XX]

The following additional DIDs (non-exhaustive list) may be listed in the CDRL depending on the acquirer’s requirements to obtain intellectual property rights. These are optional with respect to this framework, and the acquirer’s contracting policy and certification guidance documents should be sought for specific requirements regarding deliverables.

- System Development Specification
- Sub-System Design Document
- Software Requirements Specification
- Software Design Document
- Source Code Repository
- Executable Code Repository
- Toolset Repository
- Software Lifecycle Data Repository
- Software Verification Plan
- Software Verification Plan
- Software Verification Results
- System Verification Results

Note that where contractor format documents were shown during the tender evaluation to meet the intent of the contract DIDs, then the contractor format documents may be listed instead.

Contract Data Requirements List (CDRL)

Table 8 provides an example CDRL listing contract data items related to the above SOW clauses.

CDRL	Title	Delivery Schedule	
		Drafts (Preliminary)	Final
#1	Software Development Plan	ED	ED+60
#2	System and Software Architectural Assurance Document	System PDR -14, System CDR-14 Software FQTRR-14	GTRR)14
#3	Software Assurance Summary	Software PDR-14, Software CDR-14 Software FQTRR -14	GTRR-14
#4	Software Configuration Index / Software Version Description	Software FQTRR-14 GTRR-14	FTRR-14
#5	Safety Case	System PDR-14 Software PDR-14 System CDR-14 Software CDR-14 Software FQTRR-14 GTRR-14	FTRR-14

ED = Effective Date, PDR = Preliminary Design Review, CDR = Critical Design Review, FQTRR = Formal Qualification Test Readiness Review, GTRR = Ground Test Readiness Review, FTRR = Flight Test Readiness Review
 Note that the lead and lag timeframes specified above are notional for illustration purposes and may be adjusted pre-contract signature by the contract preparer to suit specific project lifecycle requirements.

Table 8: Contract Data Requirements List (CDRL) Details

Table 9 provides an example CDRL listing for additional documents that may be acquired depending on intellectual property requirements, and are provided to show the preparation and delivery lifecycle relationship to the CDRL at Table 8.

CDRL	Title	Delivery Schedule	
		Drafts	Final
#A	System Development Specification	System PDR-14, System CDR-14, System FQTRR-14, GTRR-14	FTRR-14
#B	Sub-System Design Document	System PDR-14, System CDR-14, System FQTRR-14, GTRR-14	FTRR-14
#C	Software Requirements Specification	Software PDR-14 Software CDR-14 Software FQTRR -14	GTRR-14
#D	Software Design Document	Software CDR-14 Software FQTRR -14	GTRR-14
#E	Software Verification Plan	Software PDR-14 Software CDR-14	Software FQTRR -14
#F	Software Verification Results	GTRR-14	FTRR-14
#G	System Verification Plan	System PDR-14 System CDR-14	System FQTRR-14
#H	System Verification Results	GTRR-14, FTRR-14	Final Delivery
#I	Source Code Repository Executable Code Repository Toolset Repository Software Lifecycle Data Repository	N/A	Final Delivery

Table 9: Additional Contract Data Requirements List (CDRL) Details (Project Specific)

The DIDs for these CDRL line items have not been provided in this paper, and are the subject of ongoing work. However, the SOR and SOW clauses proposed above are suggestive of the content of the CDRL DIDs of Table 8. The key focus of these documents is that the architectural assurance, satisfaction of attributes and tolerability of limitations in evidence should be presented in addition to references to the raw evidence. The safety case will sensibly include a summary of the overall argument and evidence in this regard, and should include traceability to the ASAL/CSAL/ESAL framework as necessary. The software aspects of the safety case might also take into consideration related work such as the guidance regarding evidence selection, evidence deficits, and assurance cases provided at [SSEI09].

With respect to the CDRL DIDs at Table 9, for the purposes of understanding this paper, they may be considered to be consistent with artefacts specified by standards such as RTCA/DO-178B [DO-178B], MIL-STD-492, and other similar standards.

5.5 Contract Negotiation

The contract negotiations phase of the open tender lifecycle is where fine tuning of the contract is made based on the preferred tenderer's tender evaluation results, with an aim to ensure an acceptable contract for both acquirer and contractor. In many respects, contract negotiation is an extension of the contract preparation, as it provides the means to negotiate and fine tune the contract preparation.

The key aspects that are sought with respect to the ASAL/CSAL/ESAL framework during contract negotiation are as follows:

- agreed refinement of preliminary system architecture and software architecture to meet ASAL requirements including agreement on the extent and types of:
 - absence assurance against systematic failure modes; and
 - detection/Handling mechanisms
- agreed conceptual CSAL re-assignment for each constraint where the tenderers proposed assignment did not meet certification authority expectations;

- agreed refined proposal of methods and techniques to satisfy ESAL criteria with respect to attributes at conceptually assigned CSALs, to limit the extent to which evidence shortfalls under the contract might not be tolerable; and
- agreed mechanisms for resolving shortfalls in any of the above through the contract SOW

Agreement should be reached through refinement of tender provided artefacts, either during contract negotiations, or through the draft/final approval process for CDRLs provided under the contract. Which option is pursued is dependent on the complexity and impact of the issue, and the degree to which certainty would or would not be achieved prior to contract signature.

Resolving evidence shortfalls under the contract SOW is difficult because it is very difficult to cost for the contractor as a specific line item. The difficulty in costing it may cause some contractors to increase the contract cost dramatically as a risk mitigation against the clause being invoked. Therefore it is in both the acquirer's and contractor's interests to use contract negotiations effectively to bound so far is reasonably practical, the costs of rework by effective evaluation of proposed approaches prior to final contract negotiations and contract signature.

6 Restricted Tender Acquisition

From the ASAL/CSAL/ESAL framework implementation perspective, the Restricted Tender is really no different to the Open Tender scenario. Two or more tenderers will still be engaged, and mechanisms will still be required to establish the suitability (both in terms of architecture and evidence) of the proposed solutions. Therefore, the approach outlined in Section 5 is recommended for the Restricted Tender.

In cases where a funded Restricted Tender is used to provide funding for further development of the tender artefacts (e.g. a second phase of tender after preferred suppliers have been identified in the first phase), minor revision to the tender SOR and SOW clauses is required to reflect the revised starting point for tender artefacts and alternative funding, delivery and payment schedules. Otherwise, the level of disclosure sought should be at least that proposed in Section 4.1. Such a funded tender phase provides more mature tender artefacts and design solutions thus enabling more detailed evaluation against the ASAL/CSAL/ESAL framework. It may be used as a mean of reducing project risk for highly developmental programs.

7 Sole Source Acquisition

In many respects the Sole Source Acquisition can be the most difficult acquisition paradigm to contract for assurance. This is because the acquisition may be dominated by perceptions (from both supplier and acquirer) that the solution is already safe and fit for purpose. This is particularly the case where a version of the system is already fielded with another customer, and has some in-service history. The following sub-sections outline the key elements of implementing the ASAL, CSAL and ESAL framework for each lifecycle phase of a sole source acquisition.

7.1 Request for Proposal Preparation

The Request for Proposal (RFP) preparation lifecycle phase is where the proposal request is prepared. RFP's usually consist of a covering proposal explanation, draft terms and conditions (T&Cs), an abridged Statement of Requirement (SOR) focusing on the product to be acquired, and a Statement of Work (SOW) to detail product and evidence requirements. There are two broad cases for a Sole Source Acquisition: those where a product (and its evidence set) will already exist; and those where the basis of a sole source acquisition is due

to some more abstract factor that ties the acquisition to the sole supplier (e.g. existing intellectual property, only supplier capable of working with the nominated technology, etc).

Ideally for both cases, the proposal should be as robust a document as the Open and Restricted Tender documents already discussed in Sections 5 and 6. This is the approach recommended for implementing the ASAL/CSAL/ESAL framework for a sole source acquisition. However, it may be possible for an existing product, or one where the product and evidence set already exists, there may be pressures not to enforce differing assurance requirements onto the product. In these cases many project authorities will want proposal documentation that focuses on what the supplier has done for the product (rather than will do). This is where one of the key advantages of the ASAL/CSAL/ESAL frameworks is highlighted. The ASAL/CSAL/ESAL framework can be use retrospectively as a means of assembling existing evidence to determine:

- the extent to which the product behaviours may be appropriate for the acquirers intended application; and
- the extent to which any identified limitations in the existing evidence suite will be tolerable.

Therefore, despite the different perceptions surrounding the sole source acquisition, the sole source approach for implementing the ASAL/CSAL/ESAL framework should be holistically the same as the Open Tender and Restricted Tender. The key difference is when the product and evidence already exists, the wording of the SOW will require adjustment to capture the retrospective application of the framework, and use of it to inform the acquirer about the products behaviours and sufficiency of evidence. The following are example SOR clauses to demonstrate how this might be achieved (the changes from the Tender SOW clauses are very subtle to indicate the retrospective nature of the assessment):

Informing Architectural Suitability

The [Proposer] shall prepare a [System and Software Architecture Suitability Document] to describe how the [System Name] architecture and mechanisms for achieving fault tolerance against systematic faults meets the Architectural Safety Assurance Level (ASAL) requirements defined in [Table 2 of this paper].

The [Proposer] shall prepare a [System and Software Architecture Suitability Document] to describe how each proposed constraint (i.e. absence/detection and handling mechanism) is proposed to achieve the ASAL Architecturally Layered Fault Tolerance Requirements as defined in [Table 3 of this paper]; or be shown to provide an equivalent level of fault tolerance by alternative means.

Assurance of Constraints using Claims Assurance (CSAL)

The [Proposer] shall prepare a [Software Assurance Summary] to describe the Claims Safety Assurance Level (CSAL) proposed for each constraint described in the [System and Software Architecture Suitability Document] as per [Table 4 of this paper].

Assurance of Evidence (ESAL and Tolerability of Limitations)

Defining the Evidence

The [Proposer] shall prepare a [Software Development Plan], or equivalent document, to describe the methods and techniques that are proposed or were used throughout the software development lifecycle, including description of techniques or methods used prior to this development but for which evidence is relevant.

The [Proposer] shall prepare a [Software Development Plan], or equivalent document, to describe how all evidence, both new and existing, or produced from the application of [Proposer] proposed methods and techniques are/will be documented, stored, and retrievable.

The [Proposer] shall prepare a [Software Development Plan] to describe how CDRLs [refer list] will be produced per the schedule [at Table 8].

Assessing the Evidence

The [Proposer] shall prepare a [Software Assurance Summary] to describe the extent to which the evidence produced from the application of the [Proposer] proposed methods and techniques achieves the Evidence Safety Assurance Level (ESAL) requirements for tolerability of limitations as defined in [Table 7 to this paper]; for each attribute of each software lifecycle product [per Annex A to this paper], at the CSAL [defined per Table 5] and as described in the [System and Software Architecture Suitability Document] for each identified constraint.

The [Proposer] shall prepare a [Software Assurance Summary] to describe the means, either via provision of evidence or via access provisions to tenderer facilities and data, for the [Acquirer] to inspect or review all evidence, both new and existing, from the application of [Proposer] proposed methods and techniques for the purposes of certification evaluation by the [Acquirer].

Exemplar Elements of the Software Aspects of the Safety Case

The [Proposer] shall prepare an [Exemplar Safety Case] to show the safety assessment outcomes of the ASAL, CSAL and ESAL framework for at least one constraint in each generalised category, type or class of constraint proposed. The [Proposer] shall describe the set of categories, types or classes by which they have categorised the proposed constraints.

7.2 Proposal Response

The proposal response is the phase of the process where the proposer prepares their response documentation to the Proposal SOR and SOW clauses defined above. While the onus of this activity is on the proposer, the section will elaborate the rationale for the differences of the aforementioned SOR and SOW clauses for the existing product circumstances to assist proposers in preparing their responses.

The proposal SOW requires the preparation of a Preliminary System and Software Architecture document to meet ASAL requirements including conceptual identification of absence mechanisms and detection/handling mechanisms – i.e. constraints. The purpose of the proposal requesting this information is to permit evaluation of the extent to which the existing product is likely to achieve the objectives and benchmarks set by the ASAL/CSAL/ESAL framework. That way, if holistic safety and software architecture shortfalls are identified, their resolution can be costed into the proposal response; or dealt with prior to contract signature during contract negotiations. It should be noted that the retrospective incorporation of constraints to treat systematic failure modes is rarely straightforward, particularly when it affects architecture; and the emphasis for the proposer at this point will be to inform the acquirer of potential architectural shortfalls and how the proposer intends to overcome them.

The proposal SOW also requires the preliminary CSAL assignment for each constraint, along with the identification of evidence from the application of methods and techniques to satisfy ESAL criteria with respect to attributes at assigned CSALs (generic). The purpose of requesting this information to permit the certification authority to evaluate the existing breadth and depth of evidence with respect to the general tolerability of limitations concept. It provides the acquirer a measure of the extent to which the proposed evidence is likely to meet with acquirer evidence benchmarks. It also permits the contract authority determination during the proposal evaluation as to whether the proposal adequately costs the production of any additional evidence, and what any evidence shortfalls might notionally cost in addition to the product cost.

The final element of the proposal request is the provision of exemplar attributes satisfaction for at least one constraint in each general class of constraint. The exemplar safety case

population is intended to assist in establishing the proposers understanding of the application of the ASAL/CSAL/ESAL framework (even if retrospectively) with respect to the objectives of the framework and the benchmarks for evidence defined by the framework. It permits the certification authority to evaluate whether the proposer's product and its evidence is likely to be acceptable and the extent to which any product or evidence shortfalls (versus the tolerability of limitations) will require resolution under the contract.

7.3 Proposal Evaluation

The proposal evaluation is the phase where the proposal response is evaluated against the set of pre-determined criteria that includes measures of:

- compliance with the SOR, and SOW
- suitability of the product in terms of fitness for purposes as well as safety
- proposer cost estimates
- proposer schedule estimates
- adjusted cost and schedule estimates based on shortfalls or limitations in the proposers response

With respect to the ASAL/CSAL/ESAL framework, the proposal evaluation should examine the architectural suitability, as well as the 'tolerability of limitations' to assuring attributes using the extant evidence against ESAL criteria with respect to attributes at conceptually assigned CSALs. It should also include an evaluation of 'tolerability of limitations' to assuring attributes of attribute satisfaction in the exemplar safety case. This part of the proposal evaluation should be carried out of specialists with both system domain, and safety/software assurance experience; all of who are familiar with the fault tolerance principles that underpin the ASAL/CSAL/ESAL framework.

The aim of the proposal evaluation is establishment of the following:

- Certification Authority understanding of the proposers product and evidence, including architecture, fault tolerance and provision of assurance evidence;
- the suitability of the architecture and the level of fault tolerance it provides;
- potential treatments to architectural and fault tolerance shortfalls;
- the Tolerability of Limitations in existing evidence;
- the scope, as well as cost and schedule impost, of additional evidence required to resolve intolerable evidence deficiencies; and
- the degree to which the above factors make the acquisition suitable.

On the basis of the establishing the suitability of the product and its evidence, contract negotiations will be entered with the proposer.

7.4 Contract Preparation

The contract preparation lifecycle phase is where the contract documentation is prepared for use in contract negotiations and for contract signature. Contract documentation usually consists a set of covering contract explanation, terms and conditions (T&Cs), the Contract Statement of Requirement (SOR), and Contract Statement of Work (SOW). Suitable Contract SOR and SOW clauses have already been established in Section 5.4 on contract preparation. While minor wording changes might be required to reflect the retrospective nature of some of the evidence, the holistic intent is consistent. The following sections examine possible additional clauses necessary to contract for the ASAL/CSAL/ESAL framework for the sole source acquisition of an existing product, where a certain amount of evidence already exists.

7.4.1 Contract SOR Clauses

The contract SOR clauses are the same as for the Open Tender and Restricted Tender. Additional clauses may be added to do either of the following:

- specify specific architectural mechanisms to resolve ASAL framework and associated fault tolerance deficiencies - these will be product specific, and so it is not possible to provide templates for these.
- require that the contractor resolve the shortfalls by proposing suitable architectural mechanisms.

The inclusion of which approach is adopted will be at the discretion of the contracting authority, and dependent on the results of the proposal evaluation and contract negotiations.

7.4.2 Contract SOW Clauses

The contract SOW clauses are the same as for the Open Tender and Restricted Tender. Additional clauses may be added to do either of the following:

- specify specific architectural mechanisms work scope to resolve ASAL framework and associated fault tolerance deficiencies - these will be product specific, and so it is not possible to provide templates for these.
- require that the contractor resolve the shortfalls by proposing suitable architectural mechanisms.
- specify specific additional evidence work scope to resolve ESAL/CSAL framework deficiencies – these will be product specific, and so it is not possible to provide templates for these.

The inclusion of which approach is adopted will be at the discretion of the contracting authority, and dependent on the results of the proposal evaluation and contract negotiations.

7.5 Contract Negotiation

The contract negotiations phase is where fine tuning of the contract is made based on the proposal evaluation results. The aim is to ensure an acceptable contract to both acquirer and contractor. In many respects, contract negotiation is an extension of the contract preparation, as it provides the means to negotiate and fine tune the contract preparation.

The key aspects that are sought with respect the ASAL/CSAL/ESAL framework during contract negotiation are as follows:

- agreed refinement of the established system architecture and software architecture to meet ASAL requirements including agreement on the extent and types of:
 - Absence assurance against systematic failure modes; and
 - Detection/Handling mechanisms
- agreed conceptual CSAL assignment/re-assignment for each constraint where the proposers assignment did not meet certification authority expectations;
- agreed supplementation to previously applied methods and techniques to provide additional evidence to shore up shortfalls in satisfaction of ESAL criteria with respect to attributes at conceptually assigned CSALs; and
- agreed means for resolving shortfalls in any of the above through the contract SOW

Agreement should be reached through refinement of proposer provided artefacts, either during contract negotiations, or through the draft/final approval process for CDRLs provided under the contract. Which option is pursued is dependent on the complexity and impact of

the issue, and the degree to which certainty would or would not be achieved prior to contract signature.

Resolving evidence shortfalls under the contract SOW is difficult because it is very difficult to cost for the contractor as a specific line item. The difficulty in costing it may cause some contractors to increase the contract cost dramatically as a risk mitigation against the clause being invoked. Therefore it is in both the acquirer and contractors interests to use contract negotiations effectively to bound so far is reasonably practical, the costs of rework by effective evaluation of proposed approaches pre-contract signature.

8 Contract Execution

The contract execution phase of the project lifecycle is where the onus is now on the contractor to develop an architecture and body of evidence inline with the ASAL/CSAL/ESAL framework benchmarks.

Focus will now move to progressively establishing achievement of the objectives of the ASAL/CSAL/ESAL framework over the project lifecycle. The recommended approach is through ongoing visibility through a series of systems engineering reviews, such as Design Reviews (Conceptual, Preliminary, Critical at System and Sub-system levels), and through progressive delivery of evidence via drafts at these reviews. Table 8 provided an example of how this might be achieved.

The key goals of the certification authority during the contract execution will be early visibility of potential shortfalls against the ASAL/CSAL/ESAL framework such that these can be addressed in the most cost effective manner for the contractor under the already established terms of the contract. The following types of shortfalls should be monitored:

- Evidence shortfalls that inform product suitability against ASAL benchmarks
- Product shortfalls against ASAL benchmarks
- Shortfalls against CSAL benchmarks, leading to limitations in ESAL evidence scope
- Evidence shortfalls against ESAL benchmarks

The key goals of the contractor during the contract execution will be to achieve contract milestones within costs and schedule constraints, while meeting the requirements of the contract SOR and SOW. Early visibility of prospective shortfalls will be essential to keeping the project within cost and schedule constraints. It is recommended that the contractor employ specialists with safety and software assurance skills sets that understand fault tolerance and the principles on which the ASAL/CSAL/ESAL framework is based, in order to minimise potential of ASAL/CSAL/ESAL framework ignorance based rework. These specialists are also recommended to have direct line of influence to the contractor project manager in order to ensure technical assurance aspects of the project are held in regard during project management and resourcing decisions.

9 Evaluation

The approach proposed in this paper for implementing ASAL/CSAL/ESAL framework in the open tender, restricted tender and sole source paradigms, including the definition of Tender and Contract SOR and SOW clauses, have been established analytically by drawing upon observations from and the author's prior experience with Australian Defence Force project contracts and tenders. The proposed implementation is yet to be subject to empirical evaluation, and this is proposed in future phases of this work. As results of the evaluation cannot be summarised as yet, this paper will instead propose how the evaluation is intended to be conducted.

The true validation of the approach proposed in this paper for contracting for the ASAL/CSAL/ESAL assurance framework would be to actually contract one or more real

projects to this framework, and evaluate the effects of this contracting approach against historical projects that used other approaches for contracting for assurance. A real project would provide the most representative indication if the contracting approach drives supplier and acquirer behaviours in a way that is consistent with the intent of the ASAL/CSAL/ESAL framework. However, it is unlikely that any real program would be willing to undertake such an endeavour without some preliminary validation evidence first. Therefore, the initial evaluate of this work needs to be by alternative means to application to a real project.

The following evaluation approach is proposed (Table 10):

Method	Agency or Project	Evaluation Approach / Objective
Peer Review	Research staff, project coordination, project support staff from organisations such as Australian, United Kingdom Defence research establishments (e.g. DSTO, DSTL), and technical specialist contractors working for Defence organisations.	Qualitative evaluation of more detailed questions derived from the following high level questions: Are the motivations for the contracting approach representative of historical project issues? Does the proposed approach provide a possible solution to the historical issues? Effectiveness, limitations, gaps? Are their fundamental assumptions of the contracting approach while might be invalid in practice? Would this contracting approach lead to the acceptance of acceptable designs and rejection of unacceptable designs without undue cost and schedule impost on the contract?
Survey Review including workshops	Project managers, contract officers of suppliers and acquirers including the Defence Materiel Organisation (DMO) (Australian Department of Defence), the United Kingdom Ministry of Defence (UK MoD), and Defence Contractors from Australian, UK and USA.	Detailed qualitative evaluation via structured question set and workshop facilitation of questions (of which the survey/workshops will provide substantial elaboration of the above questions)
Review of Historical Project Contracts Evidence /	Historical Australian Defence Force projects involving software development of aviation systems	Quantitative evaluation of historical evidence to establish? What are the historical project issues regarding contracting for assurance? What are they? What are their likely causes? Are these issues traceable and consistent with the motivations for contracting approach proposed in this paper? If this contracting approach would have been applied at the outset of the project, would in retrospect it appear to help? If the project encountered significant assurance issues, would the retrospective application of this contracting approach help?
Anti-hypothesis Evaluation	All previous evaluation evidence.	Establishment of anti-hypothesis and assessment of previous evaluations for presence of evidence supporting anti-hypothesis (i.e. show that the opposite of the underlying hypothesis can be correlated to known problems – and thus that this proposed approach will not be prone to the same issues)

Table 10: Evaluation Proposal

As this work proposed in this paper is tightly coupled to the underlying ASAL/CSAL/ESAL framework, the evaluation of the contracting approach is not independent of the evaluation of

the framework. In addition to the above evaluation approaches, the ASAL/CSAL/ESAL framework will be evaluated by application to:

- a constructed example by the authors; and
- retrospective application to several real system developments as an approach for structuring and ordering the examination (via audit) of evidence produced within the development.

Pending the conduct, analysis of the results of the proposed evaluation, the longer term aspiration is to identify a real project to which to apply this framework to.

10 Summary

This paper has examined the means by which the proposed ASAL/CSAL/ESAL assurance framework might be contracted for in the acquisition or modification of military aviation systems. Contracting paradigms including the preparation and execution of open tenders, restricted tenders, sole source contracts have been examined to determine how contracting for the proposed assurance framework might differ between these circumstances. The paper presents draft examples of Tender SOR, Tender SOW, Contract SOR and Contract SOW clauses to implement the ASAL/CSAL/ESAL framework. Furthermore, guidance is provided on contracting for architectural, claims and evidence assurance across the tender/contract lifecycle, including project definition and approval, tender preparation, tender responses, tender evaluation, contract preparation, contract negotiation and contract execution.

The approach chosen to implement the ASAL/CSAL/ESAL framework is based upon establishing the minimum necessary understanding of architectural fault tolerance and assurance during the tender/proposal phase, to enable effective tender/proposal evaluation, and to support bound-able contract negotiations and contract signature.

An approach to evaluating the contracting approach for the ASAL/CSAL/ESAL framework has been proposed to provide validation evidence to support future application to a real project.

11 References

The following documents, papers and publications are referenced throughout this paper. A number of these documents are not available in the public domain for propriety or confidentiality reasons. Readers wishing to seek further information should direct their queries to the author of this paper, or the relevant standards body.

[14CFR] Title 14 *Aeronautical and Space*, Code of Federal Regulations.

[14CFR21] Title 14 Aeronautical and Space, Code of Federal Regulations Chapter I Federal Aviation Administration, Department of Transportation, Subchapter C – Aircraft, Part 21 *Certification Procedures for Products and Parts*

[14CFR25] Title 14 Aeronautical and Space, Code of Federal Regulations Chapter I Federal Aviation Administration, Department of Transportation, Subchapter C – Aircraft, Part 25 *Airworthiness Standards: Transport Category Airplanes*

[AAP1000-D] Australian Air Publication (AAP) AAP1000-D *Australian Air Power Manual*, Australian Department of Defence

[AAP7001.054] Australian Air Publication (AAP) 7001.054 *Airworthiness Design Requirements Manual*, Directorate General Technical Airworthiness, Australian Defence Force

[ANA20] Australian Government, *Air Navigation Act*, 1920.

- [ASDEFCON] Australian Government, Department of Defence, *Australian Standard for Defence Contracting (ASDEFCON) (Strategic Materiel) Version 2.3*, Defence Materiel Organisation Procurement and Contracting, Effective 01 Oct 2009.
- [CAA88] Australian Government, *Civil Aviation Act*, 1988.
- [DO178B] RTCA Inc., *RTCA/DO-178B: Software Considerations in Airborne Systems and Equipment Certification*, Washington D.C.: RTCA Inc., 1992.
- [JTM07] D. Jackson, M. Thomas, L Millet, Editors, *Software for Dependable Systems: Sufficient Evidence?*, Committee of Certifiably Dependable Software Systems, National Research Council, National Academy of Sciences, USA, 2007.
- [McD07] J.A. McDermid, *Risk, Uncertainty, Software and Professional Ethics*, 20 August 2007.
- [McK06] J. McDermid, T. Kelly, *Software in Safety Critical Systems: Achievement and Prediction*, Nuclear Future, Volume 03, No. 03, 2006.
- [NTS06] National Transportation Safety Board, *Safety Report on the Treatment of Safety-Critical Systems in Transport Airplanes*, Safety Report NTSB/SR-06/02, Washington, D.C., USA, 2006.
- [OPS2-2] Defence Instruction General (DI(G)) Operations (OPS) 2-2 *ADF Airworthiness Management*.
- [ReM10] D.W. Reinhardt, J.A. McDermid, *Assuring Against Systematic Faults Using Architecture and Fault Tolerance in Aviation Systems*, presented at the Improving Systems and Software Engineering Conference (ISSEC), Aug 2010.
- [RMc10] D.W Reinhardt, J.A. McDermid, *Assurance of Claims and Evidence for Aviation Systems*, presented at the 5th IET Conference, Oct 2010.
- [SSEI09] R. Hawkins, J. McDermid, Software Systems Engineering Initiative, SSEI-TR-0000041, *Software Safety Evidence Selection and Assurance*, Issue 1, University of York, October 2009.
- [Wea03] R.A. Weaver, *The Safety of Software – Constructing and Assuring Arguments*, PhD Thesis, Department of Computer Science, University of York, 2003.

ANNEX A - ATTRIBUTES OF SOFTWARE LIFECYCLE PRODUCTS

Specified Constraint Level Requirements

(specified at the level of the architectural constraint, and at the level at which requirements are allocated to software)

Attribute	Impact of NOT Satisfying	CSAL3	CSAL2	CSAL1	CSAL0
Developed and Defined	Specified Constraint Level Requirements for constraint {constraint} do not exist – therefore there is no basis for the relevant behaviour existing in the software	Intolerable	Intolerable	Intolerable	Tolerable
Valid – Accurate / Consistent / Complete	Specified Constraint Level Requirements for constraint {constraint} exists but the specification of the constraint is invalid – therefore, there is potential for other lifecycle products or translations to refine or implement the behaviour erroneously	Intolerable	Constrained	Constrained	Tolerable
Unambiguous / Precise	Specified Constraint Level Requirements for constraint {constraint} exists but the specification of the behaviour is ambiguous and/or imprecise – therefore, there is potential for other lifecycle products or translations to misinterpret the constraint	Intolerable	Constrained	Constrained	Tolerable
Satisfiable / Verifiable	Specified Constraint Level Requirements for constraint {constraint} exists but the behaviour cannot be verified (analytically or empirically) – therefore verification evidence for the constraint will not exist or be irrelevant	Intolerable	Constrained	Constrained	Tolerable
Compatible with Target Computer	Specified Constraint Level Requirements for the constraint {constraint} exists, but the constraint is not compatible with the target computer – therefore, the specification of the constraint is unsatisfiable and additional behaviours that violate the constraint may be initiated from the target computer	Intolerable	Constrained	Constrained	Tolerable
Traceable to Lower Level Requirements (<i>Refined Requirements or Low Level Requirements</i>)	Specified Constraint Level Requirements for the constraint {constraint} exists, but there is no traceability to a lower level refinement of the behaviour – therefore, there is no traceable basis for the refinement of the relevant Specified Constraint Level Requirements existing in the software design or code	Intolerable	Intolerable	Tolerable	Tolerable
Inadequacies in Specified Constraint Level Requirements are identified and resolved	Compliance, robustness, traceability and verification may identify inadequacies in Specified Constraint Level Requirements – therefore the behaviours implemented by the software may not be consistent with the constraint	Intolerable	Intolerable	Constrained	Tolerable

Refined Abstract Level Requirements (optional in totality)

(refined from Specified Constraint Level Requirements, while still being abstract from Low Level Requirements, and used to provide a means making claims from evidence that cannot be produced directly against Specified Constraint Level Requirements or Low Level Requirements)

Attribute	Impact of NOT Satisfying	CSAL3	CSAL2	CSAL1	CSAL0
Developed and Defined	Refined Abstract Level Requirement for constraint {constraint} does not exist – therefore there is no basis for the relevant behaviour existing in the software	Intolerable	Intolerable	Tolerable	Tolerable
Traceable to Higher Level Requirements <i>(Specified Constraint Level Requirements or high level Refined Abstract Level Requirements)</i>	Refined Abstract Level Requirements exist, but there is no traceability to the higher level Requirements associated with the constraint {constraint} – therefore, the behaviours specified by this Refined Abstract Level Requirements may not be consistent with the constraint	Intolerable	Intolerable	Tolerable	Tolerable
Valid – Accurate, Consistent, Complete	Refined Abstract Level Requirement for constraint {constraint} exists but the specification of the constraint is invalid – therefore, there is potential for other lifecycle products or translations to refine or implement the behaviour erroneously	Intolerable	Constrained	Tolerable	Tolerable
Satisfiable / Verifiable	Refined Abstract Level Requirement for constraint {constraint} exists but the behaviour cannot be verified (analytically or empirically) – therefore verification evidence for the constraint will not exist or be invalid	Intolerable	Constrained	Tolerable	Tolerable
Unambiguous / Precise	Refined Abstract Level Requirements for constraint {constraint} exists but the specification of the behaviour is ambiguous and/or imprecise – therefore, there is potential for other lifecycle products or translations to misinterpret the constraint	Intolerable	Constrained	Tolerable	Tolerable
Compatible with Target Computer	Refined Abstract Level Requirement for the constraint {constraint} exists, but the constraint is not compatible with the target computer – therefore, the specification of the constraint is unsatisfiable and additional behaviours that violate the constraint may be initiated from the target computer	Intolerable	Constrained	Tolerable	Tolerable
Traceable to Lower Level Requirements <i>(lower level Refined Abstract Level Requirements or Low Level Requirements)</i>	Refined Abstract Level Requirement for the constraint {constraint} exists, but there is no traceability to a lower level refinement of the behaviour – therefore, there is no basis for the refinement of the relevant Abstract Level Requirement existing in the software design	Intolerable	Intolerable	Tolerable	Tolerable
Compliant with Higher Level Requirements <i>(Specified Constraint Level Requirements or high level Refined Abstract Level Requirements)</i>	Refined Abstract Level Requirements exist for the constraint {constraint}, but this abstraction of requirements are not compliant with the Higher Level Requirements – therefore, the behaviours specified by the Refined Abstract Level Requirements are not consistent with the constraint	Intolerable	Constrained	Tolerable	Tolerable

Attribute	Impact of NOT Satisfying	CSAL3	CSAL2	CSAL1	CSAL0
Robust with Higher Level Requirements (<i>Specified Constraint Level Requirements or high level Refined Abstract Level Requirements</i>)	Refined Abstract Level Requirements exist for the constraint {constraint}, but this abstraction of requirements is not robust with the Higher Level Requirements – therefore, the behaviours specified by the Refined Abstract Level Requirements may not be resilient to sources of faults that might violate the constraint	Intolerable	Constrained	Tolerable	Tolerable
Inadequacies in Refined Abstract Level Requirements are identified and resolved	Compliance, robustness, traceability and verification may identify inadequacies in Refined Abstract Level Requirements – therefore the behaviours implemented by the software may not be consistent with the constraint	Intolerable	Constrained	Tolerable	Tolerable

Low Level Requirements

(specified at a level that no additional refinement is required to develop source code and that all behaviours of the source code are described by the Low Level Requirements)

Attribute	Impact of NOT Satisfying	CSAL3	CSAL2	CSAL1	CSAL0
Developed and Defined	Low Level Requirements for constraint {constraint} do not exist – therefore there is no basis for relevant behaviour existing in software	Intolerable	Intolerable	Tolerable	Tolerable
Traceable to Higher Level Requirements (Specified Constraint Level or Refined Abstract Level)	Low Level Requirements exist, but there is no traceability to Higher Level Requirements associated with {constraint} – therefore, behaviours specified by Low Level Requirements may not be consistent with the constraint	Intolerable	Intolerable	Tolerable	Tolerable
Valid – Accurate / Consistent / Complete	Low Level Requirements for constraint {constraint} exist but the specification of the constraint is invalid – therefore, there is potential for other lifecycle products or translations to refine or implement the behaviour erroneously	Intolerable	Constrained	Tolerable	Tolerable
Satisfiable / Verifiable	Low Level Requirements for constraint {constraint} exist but the behaviour cannot be verified (analytically or empirically) – therefore verification evidence for the refinement of the constraint will not exist or be invalid	Intolerable	Constrained	Tolerable	Tolerable
Unambiguous / Precise	Low Level Requirements for constraint {constraint} exist but the specification of the behaviour is ambiguous or imprecise – therefore, there is potential for other lifecycle products or translations to misinterpret the constraint	Intolerable	Constrained	Tolerable	Tolerable
Compatible with Target Computer	Low Level Requirements for the constraint {constraint} exist, but the constraint is not compatible with the target computer – therefore, therefore, the specification of the constraint is unsatisfiable and additional behaviours that violate the constraint may be initiated from the target computer	Intolerable	Constrained	Tolerable	Tolerable
Traceable to Source Code	Low Level Requirements for the constraint {constraint} exist, but there is no traceability to an implementation level refinement of the behaviour – therefore, there is no basis for the refinement of the Low Level Requirements existing in the software source code	Intolerable	Intolerable	Tolerable	Tolerable
Compliant with Higher Level Requirements (Specific Constraint Level Requirements or Refined Abstract Level Requirements)	Low Level Requirement exist for the constraint {constraint}, but the low level requirements are not compliant with the Higher Level Requirements – therefore, the behaviours specified by the Low Level Requirements are not consistent with the constraint	Intolerable	Constrained	Tolerable	Tolerable

Attribute	Impact of NOT Satisfying	CSAL3	CSAL2	CSAL1	CSAL0
Robust with Higher Level Requirements (<i>Specific Constraint Level Requirements or Refined Abstract Level Requirements</i>)	Low Level Requirements exist for the constraint {constraint}, but are not robust with the Higher Level Requirements – therefore, therefore, the behaviours specified by the Low Level Requirements may not be resilient to sources of faults that might violate the constraint	Intolerable	Constrained	Tolerable	Tolerable
Inadequacies in Low Level Requirements are identified and resolved	Compliance, robustness, traceability and verification may identify inadequacies in Low Level Requirements – therefore the behaviours implemented by the software may not be consistent with the constraint	Intolerable	Constrained	Tolerable	Tolerable

Software Source Code

(compiler or assembler readable code)

Attribute	Impact of NOT Satisfying	CSAL3	CSAL2	CSAL1	CSAL0
Developed and Defined	Source Code for constraint {constraint} does not exist – therefore no basis for the relevant behaviour existing in the software	Intolerable	Intolerable	Intolerable	Tolerable
Traceable to Low Level Requirements	Source Codes exists, but there is no traceability to the Low Level Requirements associated with the constraint {constraint} – therefore, the behaviours implemented by the Source Code may not be consistent with the constraint	Intolerable	Intolerable	Tolerable	Tolerable
Valid – Accurate / Consistent / Complete	Source Code for the constraint {constraint} exist but the implementation of the constraint is incorrect – therefore, the executable object code will contain an erroneous behaviour	Intolerable	Intolerable	Constrained	Tolerable
Satisfiable / Verifiable	Source Code for constraint {constraint} exists but the behaviour cannot be verified (analytically or empirically) – therefore verification evidence for the implementation of the constraint will not exist or be invalid	Intolerable	Intolerable	Constrained	Tolerable
Unambiguous / Precise	Source Code for constraint {constraint} exists but the implementation of the behaviour is ambiguous or imprecise – therefore, there is potential for implementation of other software components or compiler/linker translations to misinterpret the constraint and introduce vulnerabilities that violate the constraint	Intolerable	Intolerable	Constrained	Tolerable
Compatible with Target Computer	Source Code for the constraint {constraint} exists, but the constraint is not compatible with the target computer – therefore, the implementation of the constraint is invalid and additional behaviours that violate the constraint may be initiated from the target computer	Intolerable	Constrained	Constrained	Tolerable
Traceable to Executable Object Code	Source Code for the constraint {constraint} exists, but there is no traceability to a target computer level refinement of the behaviour in object code – therefore, there is no basis for the complete refinement of the relevant Source Code existing in the Executable Object Code	Intolerable	Constrained	Tolerable	Tolerable
Compliant with Low Level Requirements	Source Code exists for the constraint {constraint}, but the Source Code is not compliant with the Low Level Requirements – therefore, the behaviours implemented by the Source Code are not consistent with the constraint	Intolerable	Constrained	Tolerable	Tolerable
Robust with Low Level Requirements	Source Code exist for the constraint {constraint}, but are not robust with the Higher Level Requirements – therefore, the behaviours implemented by the software may not be resilient to sources of faults that might violate the constraint	Intolerable	Constrained	Constrained	Tolerable

Attribute	Impact of NOT Satisfying	CSAL3	CSAL2	CSAL1	CSAL0
Inadequacies in Source Code are identified and resolved	Compliance, robustness, traceability and verification may identify inadequacies in Source Code – therefore the behaviours implemented by the software may not be consistent with the constraint	Intolerable	Intolerable	Constrained	Tolerable

Executable Object Code

(target computer readable binary code)

Attribute	Impact of NOT Satisfying	CSAL3	CSAL2	CSAL1	CSAL0
Produced	Executable Object Code for constraint {constraint} does not exist – therefore no basis for the refinement of the relevant behaviours of the constraint existing in the software	Intolerable	Intolerable	Intolerable	Tolerable
Integrated onto Target Computer	Executable Object Code for constraint {constraint} exists, but it didn't integrate/load onto the target computer – therefore, there is no basis for the refinement of the relevant behaviours of the constraint existing in the software on the target computer	Intolerable	Intolerable	Intolerable	Tolerable
Compatible with Target Computer	Executable Object Code for the constraint {constraint} exists, but the constraint is not compatible with the target computer – therefore, the implementation of the constraint is invalid and additional behaviours that violate the constraint may be initiated from the target computer	Intolerable	Intolerable	Constrained	Tolerable
Traceable to Source Code	Executable Object Code exists, but there is no traceability to the Source Code associated with the constraint {constraint} – therefore, the behaviours implemented by the Executable Object Code may not be consistent with the constraint	Intolerable	Constrained	Tolerable	Tolerable
Compliant with Specified Constraint Level Requirements	Executable Object Code exists for the constraint {constraint}, but the Object Code is not compliant with the Specified Constraint Level Requirements – therefore, the behaviours implemented by the software are not consistent with the constraint	Intolerable	Intolerable	Constrained	Tolerable
Robust with Specified Constraint Level Requirements	Executable Object Code exists for the constraint {constraint}, but the Object Code is not robust with the Specified Constraint Level Requirements – therefore, the behaviours implemented by the software may not be resilient to sources of faults that might violate the constraint	Intolerable	Intolerable	Constrained	Tolerable
Verification Coverage of Specified Constraint Level Requirements	Executable Object Code exists for the constraint {constraint}, but the Object Code is not verified against all applicable Specified Constraint Level Requirements – therefore, the behaviours implemented by the software may not be consistent with the constraint	Intolerable	Intolerable	Intolerable	Tolerable
Compliant with Refined Abstract Level Requirements	Executable Object Code exists for the constraint {constraint}, but the Object Code is not compliant with the Refined Abstract Level Requirements – therefore, the behaviours implemented by the software are not consistent with the constraint	Intolerable	Constrained	Tolerable	Tolerable
Robust with Refined Abstract Level Requirements	Executable Object Code exists for the constraint {constraint}, but the Object Code is not robust with the Refined Abstract Level Requirements – therefore, the behaviours implemented by the software may not be resilient to sources of faults that might violate the constraint	Intolerable	Constrained	Tolerable	Tolerable

Attribute	Impact of NOT Satisfying	CSAL3	CSAL2	CSAL1	CSAL0
Verification Coverage of Refined Abstract Level Requirements	Executable Object Code exists for the constraint {constraint}, but the Object Code is not verified against all applicable Refined Abstract Level Requirements – therefore, the behaviours implemented by the software may not be consistent with the constraint	Intolerable	Constrained	Tolerable	Tolerable
Compliant with Low Level Requirements	Executable Object Code exists for the constraint {constraint}, but the Object Code is not compliant with the Low Level Requirements – therefore, the behaviours implemented by the software are not consistent with the constraint	Intolerable	Constrained	Tolerable	Tolerable
Robust with Low Level Requirements	Executable Object Code exists for the constraint {constraint}, but the Object Code is not robust with the Low Level Requirements – therefore, the behaviours implemented by the software may not be resilient to sources of faults that might violate the constraint	Intolerable	Constrained	Tolerable	Tolerable
Verification Coverage of Low Level Requirements	Executable Object Code exists for the constraint {constraint}, but the Object Code is not verified against all applicable Low Level Requirements – therefore, the behaviours implemented by the software may not be consistent with the constraint	Intolerable	Constrained	Tolerable	Tolerable
Compliant with Source Code	Executable Object Code exists for the constraint {constraint}, but the Object Code is not compliant with the Source Code – therefore, the behaviours implemented by the software are not consistent with the constraint	Intolerable	Constrained	Tolerable	Tolerable
Robust with Source Code	Executable Object Code exists for the constraint {constraint}, but the Object Code is not robust with the Source Code – therefore, the behaviours implemented by the software may not be resilient to sources of faults that might violate the constraint	Intolerable	Constrained	Tolerable	Tolerable
Verification Coverage of Source Code Structure	Source Code exists for the constraint {constraint}, but the verification has not exercised all behaviours of the Source Code relevant to the constraint – therefore, there may be additional behaviours of the source code which violate the constraint	Intolerable	Constrained	Tolerable	Tolerable
Verification Coverage of Executable Object Code Structure	Executable Object Code exists for the constraint {constraint}, but the verification has not exercised all behaviours of the Executable Object Code relevant to the constraint – therefore, there may be additional behaviours of the Executable Object Code which violate the constraint	Intolerable	Constrained	Tolerable	Tolerable
Inadequacies in Executable Object Code are identified and resolved	Compliance, robustness, traceability and verification may identify inadequacies in Executable Object Code – therefore the behaviours implemented by the software may not be consistent or complete with the constraint	Intolerable	Constrained	Tolerable	Tolerable

