Certification Authorities Software Team (CAST)

Position Paper CAST-9

Considerations for Evaluating Safety Engineering Approaches to Software Assurance

Completed January, 2002

Considerations for Evaluating Safety Engineering Approaches to Software Assurance

1. Introduction.

Safety engineering approaches for software assurance (SEASA) can be applied to software with the purpose of satisfying the requirements that are mandated by aviation regulations for equipment and systems installed on aircraft. However, policy to guide certification authority engineers in their assessment of SEASA proposals as a means of compliance for the software has been limited to issue papers applied to specific projects. The purpose of this position paper is to propose guidelines for certification authorities to consider when evaluating the acceptability of SEASA and for industry organizations considering such an approach as a method for providing safety, system and software compliance assurance. It should be noted that safety engineering approaches in general are not a replacement in total for the guidelines of DO-178B/ED-12B (reference a.). Rather, safety engineering approaches can be used in conjunction with DO-178B, supplementing and/or replacing some criteria of DO-178B for some software components, using all criteria of DO-178B for some other software components, and perhaps validating software assurance criteria for yet other components. The following sections of this paper will discuss some safety engineering approaches. It also indicates which assurance objectives are always common between SEASA and DO-178B, and which DO-178B objectives can potentially be supplemented, reduced or replaced by using a SEASA.

Note: Safety engineering approaches are variously called “safety directed development (SDD),” “systems safety engineering techniques (SSET),” and “safety engineering techniques for software (SETS).” For this paper, all will be referred to by the acronym “SEASA” – safety engineering approaches for software assurance.

2. Reference Documents.

a. RTCA, Inc. document DO-178B and EUROCAE document ED-12B, “Software Considerations in Airborne Systems and Equipment Certification”, dated December 1, 1992.
b. FAA Advisory Circular (AC) 20-115B, “RTCA, Inc., Document RTCA/DO-178B”, dated January 11, 1993.
c. FAA AC 23.1309-1C, “Equipment, Systems, and Installations in Part 23 Airplanes”, dated March 12, 1999.
d. FAA AC 25.1309-1A, “System Design and Analysis”, dated June 21, 1988.

e. JAA Advisory Material-Joint (AMJ) 25.1309, “System Design and Analysis”, dated May 11, 1990.
f. JAA Temporary Guidance Leaflet (TGL) #4, “Software Guidance”, dated 1 October 1996
g. Society of Automotive Engineers (SAE) Aerospace Recommended Practice (ARP) 4754 and EUROCAE document ED-79, “Certification Considerations for Highly-Integrated or Complex Aircraft Systems,” dated November 1996.
h. SAE ARP 4761, “Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment”, dated December 1996.
i. RTCA, Inc. document DO-254 and EUROCAE document ED-80, “Design Assurance Guidance for Airborne Electronic Hardware,” dated 19 April 2000.
j. RTCA, Inc. document DO-248B and EUROCAE document ED-94B, “Final Report for Clarification of DO-178B(/ED-12B) ‘Software Considerations In Airborne Systems and Equipment Certification’”, 2001
k. CAST Position Paper CAST-5, “Alternate Means”
l. “Capability Maturity Model for Software, Version 1.1”, dated February, 1993.
m. IEEE/EIA 12207, “Software Life Cycle Processes”, dated March, 1998.

3. Background.

Reference a. is recognized by the certification authorities directly by references b. and f., and indirectly by references c. through e. as providing an acceptable means of compliance for the software aspects of certification to aviation regulations when software is installed as part of an airborne system. Software aspects are usually associated primarily with FAR and JAR XX.1301, XX.1309 (XX = 23, 25, etc.) and 33.28 (engine control systems). In fact, FAR 33.28 is the only regulation that specifically uses the term “software.”

Reference a. provides guidance for a structured, rigorous development and verification process. This rigor is based on the failure condition classification of the system functions in which the software is used. A software level is assigned, based on the worst-case failure condition category to which anomalous behavior of the software could cause or contribute. It provides software process assurance objectives and criteria for software planning, development, verification, configuration management, quality assurance and certification liaison. Verification in reference a. consists of reviews, analyses and testing conducted throughout the development process. Reviews and analyses are performed on the software data throughout the development to ensure that errors and deficiencies are not propagated from one development process to a subsequent process. Testing consists of requirements-based testing for normal range and abnormal range (robustness) at the software module level, integrated software module level and integrated hardware and software level, combined with requirements-based test coverage analysis and code structural coverage analysis. The reference a. software verification objectives and criteria were developed to ensure that the software functions as specified and to reduce the likelihood that the software will contain errors or exhibit anomalous behavior that could effect safety. The amount of structure and rigor of the software verification process is dependent on the software level as specified in reference a. for software levels A through E. The software level for a specific software application is based upon a system safety assessment process, such as that described in reference h. This determines the worst-case failure conditions of the system that could result from software anomalous behaviors caused by errors. Reference g. also provides guidance for types of safety and system engineering analyses, methods and techniques that could be performed to ensure the safety of the system, satisfy the "fail safe design techniques" of references c. through e., and comply with the regulations.

Because of the complex nature of many software applications, there are inherent difficulties in implementing and showing independence between those software components that could contribute to a failure condition affecting safety and those that could not. This is particularly the case when the software application is large, and if the code is not structured primarily for the compartmentalization of safety related elements. It may also be inherently difficult in such applications to identify a strict correspondence (traceability) between system safety requirements and the discrete components of the software application. With increasing size and complexity, safety analysis of the software becomes increasingly problematic for applications that are mainly constructed around performance and functional requirements rather than safety requirements. Therefore, aircraft certification programs have traditionally relied on software process assurance combined with a specified level of software verification to reduce the potential errors and anomalous behavior of the software. This approach has been accepted as the practical minimum requirement for showing compliance of the software components of a system, without directly addressing those safety elements inherently difficult to address in software. Industry and regulatory authorities have jointly formalized specific criteria for implementing this approach in reference a.

Note: The reader should recognize that software process assurance and verification is only one aspect, and perhaps a small part, of the overall development, verification and validation of the system and of the system installed on the aircraft. Other significant activities include: system safety assessment, system design, hardware design, hardware environmental qualification testing, system validation and verification, aircraft architecture design, laboratory bench testing, aircraft installation testing, simulation, aircraft ground testing, and aircraft flight testing.

The process assurance and verification criteria of reference a. do not provide safety assurance in and of themselves. They provide assurance that the software application adheres to the software requirements, including safety-related, functional, performance, etc. requirements. They also provide a relative measure of how much the code and its structure have been exercised, considering the assigned software level of the application. However, software requirements may or may not be focused on safety or system failure conditions that can lead to aircraft hazards, and code can be exercised with no knowledge of the system failure conditions and aircraft hazards. Rather, aircraft hazard and system failure elimination, minimization, and control have typically been addressed at the aircraft level and the system level, respectively. Thus, error reduction of the software through process assurance and verification must be combined with analysis and validation at the aircraft and system levels and recognized as minimum requirements for showing that the safety objectives for the system (and the aircraft) have been met. While this approach is recognized as good practice, it is an indirect approach to the safety aspects of the software.

Other software standards and guidance have proposed “qualification” of software developers’ process or organization as ensuring that the process or organization will produce “high quality” products. Examples of these are the Carnegie Mellon University (CMU) Software Engineering Institute’s (SEI) Capability Maturity Model (CMM), the International Standards Organization’s (ISO) 9001, part 3; and Institute of Electrical and Electronics Engineers (IEEE), Inc.’s various software standards and guidance material (e.g., IEEE/EIA 12207). While all of these standards and guidance materials have merit and are useful for many applications, reference a. has become the de facto standard guidance for process assurance of airborne systems and equipment using digital computers and software intended for application on commercial aircraft.

Some applicants have asserted that they prefer a more direct approach of tracing potential aircraft hazards and system failures to discrete software components, and then eliminating, minimizing, or controlling specific hazards and failure conditions to which the software could contribute, or providing protective mechanisms around susceptible software components, and thus directly assuring the safety of the software.

Applicants have proposed that the development of a “software safety program” integrated with the system safety program offers a direct assurance of total system safety. Such an integrated safety program would use analysis techniques derived from system safety assessment methods, combined with complete safety related software requirements, and use safety-focused software design, coding, and verification. Applicants have also proposed that this approach of using an integrated software and system safety program may justifiably allow some relief from the criteria of reference a., the structural coverage criteria and, in some cases, perhaps a lowering or reduction of other assurance criteria or software levels from current reference a. criteria.

When evaluating an applicant’s or software developer’s proposal for a safety program for software or other alternative methods, other “industry” standards and certainly other “best engineering practices” could be considered as criteria for determining the acceptability of the proposal. However, for practical purposes, such as certification authority’s familiarity with reference a.; and to expedite the certification process and to provide standardized and harmonized criteria and thus a level playing field for applicants, reference a. is most often used as the benchmark for determining the acceptability of alternative proposals for software assurance. See also reference k., the CAST paper on “Alternate Means.”

Reference g. recognizes the use of a “Safety Directed Development Concept” as “an alternative to the system development assurance methods” of that document, without further specifying what a safety directed development concept is, what its criteria are, or why it is an acceptable alternative.

Recently published hardware development assurance guidance (reference i.) also recognizes a “Safety-Specific Analysis” as an “advanced verification method,” which can be used in combination with a “functional failure path analysis” (FFPA), for providing electronic hardware design assurance for Levels A and B hardware functions in addition to the normative process and hardware data guidance of the document. According to reference i., safety-specific analysis “focuses on exposing and correcting the design errors that could adversely affect the hardware outputs from a system safety perspective.” Appendix B of reference i. provides some clarification of this method and mentions some hardware process and data needed to use this analysis method, in addition to the normative hardware process and data guidelines provided in reference i., sections 2 through 11 and Appendix A.

In the past, a specific safety approach of an airborne system developer was accepted for showing software compliance on certain systems installed during the course of specific aircraft system certification programs. The original approach was approved as a supplement to DO-178[ ], not as an alternate. However, these specific criteria have been confined to issue papers applicable only to these specific systems on specific programs and reviewed on a project-by-project basis. Credit granted by the certification authority allowed some reduction in the software structural coverage analysis criteria for some systems.

Some applicants and developers would like to see their proposed alternative safety approach accepted on a general basis, usable at their discretion and “pre-approved” for use on any of their projects and products. This has not yet been allowed by the certification authorities, except on a case-by-case basis.

4. Discussion.

The application of a safety engineering approach for software assurance (SEASA) is intended to directly address system failure conditions and aircraft hazards related to the system's software. In its essentials, SEASA is the practice of applying common systems' safety assessment, analysis and design techniques to software. It is a practice of continuing the system safety assessment into the software components, and assuring that the software meets the regulations (i.e., §§ 23.1309, 25.1309, 27.1309, 29.1309, 33.28(e)). Some commonly used system safety assessment and analysis methods are documented in references g. and h. Fail-safe design techniques are documented in references c. through e. There is much commonality and consistency between these references for methods and techniques.

Safety design techniques propose to identify all safety requirements for the aircraft, the system and its components, including hardware and software. SEASA proposes to identify all safety requirements for the software and all safety effects of the software. Then, by a method of iterative analysis and design, combined with safety-focused verification, SEASA proposes to show that those unacceptable hazards, failure conditions and anomalous behavior are prevented, mitigated or isolated and controlled, using various architectural constraints, monitors and design techniques (e.g., multiple, independent, dissimilar components performing the same function). Aircraft hazards and system failure conditions are traced through analysis to the software. Software requirements ("safety objectives" and "safety-related requirements" in the reference a. vernacular) addressing the hazards and failure conditions are specified and verified for safety related completeness criteria. Code is analyzed, with hazards and failure conditions traced to the code itself. Code is designed for risk reduction, including the prevention, mitigation, isolation and/or control of specific failure conditions. Verification is focused on the safety related code, "critical code." Like any other software development process, this SEASA process is iterative in nature with attendant feedback and refinement.

It should be noted that the specific SEASA evaluated to-date is not a complete alternative for the criteria of reference a. In fact, the majority of reference a. criteria remain criteria for this or any SEASA as well. The process assurance objectives and criteria of reference a. are simply "best industry practice," with additions for specific guidance for the aircraft certification and airborne systems domains, for assuring that the developed software is the specified software through the use of disciplined, rigorous processes and complete, correct data. This is needed for a SEASA approach in the same manner that it is needed for other software development approaches. Plans, procedures, standards, as well as processes for requirements, design, coding, integration, configuration management, quality assurance, and verification are all needed for a SEASA. Requirements-based testing and test coverage analysis are obviously critical to SEASA. Requirements completeness and assurance of safety related requirements are key elements of the

SEASA proposal. In fact, even by the admission of a specific airborne system developer, their specific SEASA requires more resources and effort than a program that just followed the guidance of reference a. for software assurance. Again, the main area that the specific SEASA was given “credit” and allowed to reduce the reference a. criteria was in the area of structural coverage. The rationale for allowing that reduction is, however, not documented, nor is the actual amount of “credit” granted known.

Structural coverage analysis is a verification activity designed to assure that the software code has been adequately exercised in order to detect otherwise hidden software errors not revealed by requirements-based (functional) testing, and to reveal unreachable code in the software application. However, structural coverage can be achieved by testing in a technically correct, but perhaps inappropriate, manner: “structural testing.” Structural testing is testing the software, mostly at the software component level, using test procedures and cases developed on the basis of the code component itself, without necessarily being conscious of safety issues, system issues, integration considerations, or even requirements issues. Unless the software requirements, design and code have been developed very rigorously, this can lead to very non-value-added and inappropriate testing. See also related discussion paper 4.3 and frequently asked question (FAQ) 3.44 in reference i.

The focus of reference a. is to conduct requirements-based testing of the software, including safety-related requirements, high-level requirements, low-level requirements, and derived requirements, at the highest level possible, measure the structural coverage achieved, and then supplement that testing by conducting lower level requirements-based tests until the structural coverage appropriate for the software level is achieved.

Testing under SEASA is focused on safety related code, while code shown to be benign (i.e., non-safety related code) may not be exercised to the appropriate software level to satisfy the structural coverage criteria of reference a.

It is unclear what other relaxation of reference a. assurance criteria may have been granted in the past or what may be possible using a SEASA. As a SEASA is used to identify and logically separate safety related code components from non-safety related code components (i.e., partitioning), it may be possible to reduce the reference a. criteria for the benign code components, as long as failures and anomalous behavior of those components cannot affect safety-related code components nor system availability or integrity, and it can be assured that any reduction will not lead to an undesirable safety affect. A SEASA proposal will likely include a partitioning strategy for using protective mechanisms for specific components and/or software operations.

5. <u>Guidelines for evaluating a safety engineering approach for software assurance (SEASA)</u>.

The following guidelines are intended to help determine the acceptability of an aircraft system applicant's or developer's proposal to use a SEASA for software assurance, software approvals and compliance to aircraft regulations:

a. <u>Integrated system, hardware and SEASA safety program</u>. The applicant (or airborne system manufacturer or software developer) should develop and implement a safety program which is specific to the airborne product, and which combines and closely integrates the safety program and development processes for the product system, its hardware components and its software components. As any safety program for software relies on the identification and tracing of all aircraft hazards and all system failure conditions through the system requirements and architecture, hardware components, software requirements, software design and code, and the elimination or control of these hazards and failure conditions, it is imperative that the safety program be developed and implemented jointly by system developers, hardware developers and software developers. Safety assessment and elimination, minimization, and control of hazards and failure conditions should be a seamless process through the system development, hardware development and software development processes. The software developers must be informed of the system safety aspects of hardware failures and hardware and software behaviors, and the system developers must be informed of the potential hardware and software contributions to system failure conditions and aircraft hazards. Safety related system components should be identified to the software developers. Successful use of a SEASA relies on an approach where software developers are familiar with, and kept current on, safety aspects of the system. For example, software developers should be knowledgeable about system variables, such as input and output signals, which have potential safety consequences for the aircraft. This knowledge should then influence the manner in which software developers design the processing of these signals as well as the verification procedures for ensuring they are processed correctly, including handling signals and inputs that may be outside the normal "tested" ranges. Similarly, the hardware developers should be informed and knowledgeable about system variables that are processed by hardware. System developers should be knowledgeable about the safety aspects of the software and the hardware. This knowledge should influence how system requirements, architecture, design, constraints, and system level verification and validation procedures are specified. The integrated safety program should be designed to assure that safety-related information (i.e., requirements, design decisions, assumptions, constraints, problem reports, change impact analysis, etc.) is provided and used effectively between the system developers, hardware developers and software developers, and those responsible for validating and verifying the system, its hardware and its software.

b. <u>Plans, procedures, and standards</u>. All reference a. software planning and procedural data needed for compliance are also needed for compliance using a SEASA. In addition, system, hardware and software plans and procedures should define the integrated safety program, with safety related activities and data clearly specified. Standards should also be written to reflect safety-related requirements and constraints on development formats, methods, and practices. For example, software development standards should contain requirements and constraints on architectural elements related to isolation, partitioning, separation, monitoring, cohesion, and coupling of safety-related software components and parameters. Standards should also drive system, hardware and software designs towards simplification and verifiability. Design simplification and verifiability will provide better assurance that all safety related software aspects have been correctly implemented.

c. <u>Safety related software requirements, design, and code</u>. All aircraft hazards, system failure conditions and safety-related requirements should be traced to the software-hardware interface, and software requirements specific to the hazards and failure conditions should be developed. Constraints on software behavior and software related system behavior should be specified. System states and state transitions should be completely specified. Each state should have a transition to a safe state specified. All states should be reachable and default transitions should be fully specified. Transitions to unsafe states should be identified by the safety assessment process and eliminated by specification of a replacement transition to a safe state. The system and software should always start in a safe state. All variables should be initialized correctly on system startup or initialization. All software components that perform safety related operations should be identified. These safety-related software components and the resources (i.e., processor, memory, clock, input/output devices, buses, etc.) should be isolated and protected from other non-safety-related components. The software architecture should be designed so that safety related components are highly cohesive and loosely coupled. Safety-related signal and data handling should be minimized. Hazard reduction design strategies and fail safe design techniques should be used to: eliminate hazards and failure conditions, minimize hazards and failure conditions, control the effects of hazards and failure conditions, or minimize the effects of hazards and failure conditions (listed by most desirable to least). Hazard reduction strategies for software-based systems include, for example:

(1) Isolation and containment (partitioning) of safety-related software and hardware components,
(2) Addition of protective features, such as safety monitoring software or hardware,
(3) Recovery schemes from hardware failures and hardware or software anomalous behavior,
(4) Addition of barriers, such as interlocks, lockouts, and lock-ins,
(5) Functional cohesion, isolation, and decoupling (partitioning),

(6) Exposure reduction,
(7) Simplification of design for increased verification and validation certainty.

d. <u>Traceability and verifiability</u>. Aircraft hazards and system failure conditions to which software can contribute, or which are controlled by software, should be traceable to related software requirements, architecture, design components and constraints. These, in turn, should be traceable to the code, and to the verification procedures and cases that verify them under normal and abnormal conditions. Traceability should be shown from the hazards and failure conditions to the source code, and verification should be conducted to the same software level criteria as stated by reference a. criteria for the same software level. It is preferred that all safety-related software requirements-based testing be performed on the target executable code loaded on the target computer in the target environment. Testing on a high fidelity simulated environment can be acceptable if evidence exists to support the validity of the environment for the specific tests. A high fidelity simulated environment is one that provides identical or similar computing resources, peripherals, interfaces and inputs and outputs such that tests conducted on the simulated environment would result in identical, valid results to tests conducted in the actual operational environment. Qualification of the simulated environment as a verification tool may be needed. For a SEASA program, it is preferred that all safety-related requirements be verified on the executable object code integrated into the target computer and environment, or high fidelity simulated environment if applicable. If identical equipment cannot be used for technical reasons, deviations from the identical equipment must be presented to and accepted by the certification authority.

e. <u>Verification</u>. A combination of both static and dynamic analyses should be specified by the applicant/developer and applied to the software. As with reference a., the emphasis of software verification should be focused on safety-related components and potential failure conditions of the software. Software reviews of requirements, design, architecture, code, integration, verification cases and procedures, and coverage analysis that would be expected under reference a., are applicable to SEASA as well. In addition, system safety analysis methods and fail-safe design techniques should be applied to the software to determine safety-related components and the safety effects of the software logic. Verification should be performed at the same level of rigor as specified by reference a. verification criteria for the same software level. Therefore, for a SEASA application, safety related requirements should be verified on the executable object code integrated into the target computer and environment. If safety related software components are properly isolated, partitioned, protected and decoupled from other software components, then structural coverage criteria of reference a. may be reduced for the non-safety related components, as is allowed by reference a. Software testing should concentrate on showing that the safety-related components perform according to the software safety requirements and that safety-related components are not adversely affected by other, non-safety-related components (verifying the integrity of the partitioning).

f. <u>Software configuration management and software quality assurance</u>. The configuration management and quality assurance criteria for a SEASA program are identical to those of reference a. and should include QA evaluation and oversight of SEASA processes as well as configuration management of all SEASA specific data items.

g. <u>Data items</u>. All software life cycle data items of reference a. should be produced for a SEASA program, with the exception of items that the certification authority has allowed a reduction or replacement to the reference a. criteria. SEASA will likely result in additional SEASA-specific data items which replace reference a.-specified data items. Examples could be: system safety assessment results for software, justification of SEASA criteria as acceptable for satisfying one or more reference a. objectives, reference a. compliance matrix, structural coverage analysis of non-safety related components. In addition, all software data items needed to support the applicant's/developer’s SEASA process are needed. This includes those needed to support the integrated safety for system, hardware and software program. Additions to the content of reference a. software data items will be needed to support a SEASA application.

h. <u>Software levels</u>. Software levels are determined by the severity of the aircraft hazards and system failure conditions to which either the software could contribute, or is intended to control. Software levels should be determined consistent with system assurance level guidance in reference a. first and then reference g. If there is any difference in the determination of the software level as determined by references a. and g., the software level determination should be discussed and agreed-to by the specific project’s certification authority. Any area of software to which SEASA will not be applied should comply with reference a. for the appropriate software level, as determined by the system safety assessment (SSA).

i. <u>Protective mechanisms and protected components</u>. Where protective mechanisms are applied, the assignment of software level for a protected, safety-related software component should be determined by the worst case failure condition of that component, assuming the protective mechanism(s) are operable. The software level assigned to each protective mechanism should be based on the worst-case failure conditions that would result from a failure of that protective mechanism in combination with the failure of the protected software component. This assignment should consider:

(1) The failure condition which could result from the failure or anomalous behavior of the protective mechanism to properly deploy and the failure or anomalous behavior of the protected component, and

(2) The worst case failure condition of an inadvertent deployment of the protective mechanism.

j. <u>Software level exceptions</u>. Exceptions to i(2) may occur where application of SEASA results in mitigation of the effect of the inadvertent deployment.

k. <u>Reference a. criteria exceptions</u>. Once the software level is assigned, all objectives of reference a. should be satisfied for the assigned software level; except where the applicant or developer has negotiated and obtained concurrence from the certification authority that their SEASA criteria, activity or data represents an equivalent means of satisfying the objective. Exceptions and equivalent means to reference a. criteria should be justified, documented, and agreed-to early in the product development with the certification authority. Analysis from the actual application of their SEASA may, in some cases, determine that additional verification is needed to augment the SEASA method. In these cases, the applicant or developer should provide appropriate verification results.

l. <u>Software Reviews</u>. Software reviews should be conducted by the certification authority (and designees, if delegated) in the same manner and to the same criteria as with a reference a. review. The approval of SEASA processes and data should not require any extraordinary compliance monitoring beyond that required for a "standard" reference a. program. However, the reviewers should be familiar with the application of system safety assessment methods and fail-safe design techniques.

m. <u>SEASA proposals</u>. An applicant or developer should present their SEASA proposal to the certification authority and gain their approval of the proposal early in the aircraft program or system life cycle to reduce the risk to the aircraft program or system project associated with non-concurrence by the certification authority. The proposal should include a reference a. compliance matrix showing which objectives will be satisfied by their SEASA criteria, activities and data, including independence and data control categories, which will be partially satisfied by their SEASA criteria, activities and data, and which will not be satisfied by their SEASA criteria, activities and data. The proposal should include a justification for any objectives that will not be fully satisfied by the SEASA proposal. Failure of the applicant or developer to submit the compliance matrix and justification would be grounds to disallow the SEASA proposal immediately.

n. <u>SEASA evaluations</u>. The certification authority should make every reasonable effort to evaluate the applicant's or developer's SEASA proposal in a timely manner, and accept the proposal, accept the proposal with conditions, or reject the proposal with rationale for its unacceptability. It is in the best interests of the certification authority, the aircraft manufacturer, the applicant, the system developer and the software developer to resolve and come to agreement on SEASA proposals early in the certification program.

6. Conclusion.

This paper presents some considerations and guidelines for certification authorities to use when evaluating applicants' proposals to use system safety assessment and engineering methods and techniques to providing software assurance for airborne systems and equipment. Although alternate approaches were considered, they must still be carefully evaluated on a case-by-case basis. The use of these approaches may be acceptable, but should be combined with a "total" system safety program and current "best-practice" software guidelines. Applicants should prepare and provide descriptions of their proposed approach and rationale for its acceptability for software assurance, which can then be evaluated by the certification authorities to determine the acceptability of their proposed alternative approach.