



**Advisory Circular
July 1999**

Software and its use in Safety Critical Systems

This publication is only advisory. It describes recommended principles for safety management of airways systems where software is used to perform safety critical functions.

Contents

Definitions	2
Guiding principles	2
Integrated system safety effort	3
Specified Integrity levels	3
Conservative system architecture and technologies	3
Process and product based safety assurance	4
References	7

The relevant regulations and Orders

This Advisory Circular should be read in conjunction with the Civil Aviation Safety Authority document "Safety Regulation of Airservices Australia and Aerodrome Rescue and Fire Fighting Service Providers - Final Draft Regulatory Arrangements and Standards - April 1996". That document is available on request from CASA at the address at the foot of this page.

Who this Advisory Circular applies to

This Advisory Circular applies to airways service providers.

Why this publication was written

This Advisory Circular provides guidelines for safety management during the acquisition, development and implementation of airways systems where software is used to perform safety critical functions. Note that throughout the text references are shown inside square brackets.

Status of this Advisory Circular

This is the first issue of CASA/AA MOU.AIRWAYS - 2(0). It remains current until re-issued, withdrawn or superseded.

For further information

Further information is available from the Airways and Airspace Standards Branch of CASA. The address details are as follows:

Post: Airways and Airspace Standards Branch
Civil Aviation Safety Authority
GPO Box 2005
CANBERRA ACT 2601

Telephone: 131757 **FAX:** 02 62171700

1. DEFINITIONS

Airways service: An air traffic service, aeronautical information service, search and rescue service, aeronautical radio navigation service, aeronautical telecommunications service, or rescue and fire fighting service.

Airways system: Facilities, equipment, personnel and procedures, which in combination, provide an airways service. (Airways systems also encompass airspace systems used for air traffic services.)

2. GUIDING PRINCIPLES FOR USE OF SOFTWARE IN SAFETY CRITICAL SYSTEMS

- 2.1 Airways systems being deployed today are not only increasingly complex and integrated, but also more and more based on the relatively new and continually evolving technology of software. It has therefore become imperative to consider safety aspects by a structured approach commencing early in the life-cycle of a system, preferably at the operational concept phase.
- 2.2 A number of professional bodies, associations and interested parties at national and international level are continually working to define 'best practice' as well as 'standards' applicable in the area of safety and software. An analysis of this effort reveals certain high level common principles present in most initiatives (forums, standards, articles, papers, etc.).
- 2.3 The guidelines in this CAAP have been developed to outline those common principles, as CASA has adopted a non-prescriptive approach to regulating airways service providers.
- 2.4 The principles apply equally across different application domains irrespective of the specific project organisation, engineering or management standards and practices applied.
- 2.5 They can therefore be considered as guiding principles which should permeate the acquisition, development and implementation of systems where software is the technology of choice for safety critical functions.
- 2.6 The principles are:
 - **Integrated Systems Safety**
 - **Specified Integrity Levels**
 - **Conservative System Architecture and Technologies**
 - **Process and Product based Safety Assurance**

Note that inasmuch as these guidelines discuss software in the context of system and safety issues, the subjects covered may well apply to other domains.

3. INTEGRATED SYSTEM SAFETY

3.1 PRINCIPLE

- 3.1.1 **Taking a holistic approach, system safety should integrate and cut across all disciplines within a controlled and planned development environment and through all life cycle**

phases. There should be no stand alone and unconnected safety activity, including software technology [5, 12, 15, 17].

3.1.2 Agencies or individuals implementing these type of systems should take full advantage of standard system safety techniques, apply them within the software discipline and not focus solely on increasing software reliability and availability [5].

3.2 RATIONALE

3.2.1 Safety is not a property of individual system components, but rather of the working together of those components as an integrated whole where the system comprises people, procedures and equipment [5, 10].

3.2.2 System safety is about preventing accidents by building-in safety through the application of management and engineering techniques during all life cycle phases [14].

3.2.3 System safety provides the highest benefits when applied from early life-cycle phases (i.e. operational concept, early design) [5, 14].

3.2.4 The issue of whether the requirements themselves are safe is as important as whether the implemented system satisfies the requirements [5].

4. SPECIFIED INTEGRITY LEVELS

4.1 PRINCIPLE

4.1.1 Software safety assurance should be based either on the concept of specified Systematic Safety Integrity Levels or Software Levels. Each level should establish the set of processes and effort to be applied during development as well as the applicable techniques. Levels should be linked to the risk acceptability classification criteria of the project [4, 11, 15].

4.2 RATIONALE

4.2.1 Functions of different criticality are implemented in software.

5. CONSERVATIVE SYSTEM ARCHITECTURE AND TECHNOLOGIES

5.1 PRINCIPLE

5.1.1 The system architecture should take into account current limitations in achieving and assessing software reliability [3, 8].

5.1.2 Safety should not be compromised by limitations of the implementing technology (e.g. because a certain integrity cannot be reached) [16].

5.1.3 Where software design or design-process characteristics are utilised to justify the safety of a system (e.g. safety case) the information should be based on actual measurable levels of design dependability [8].

5.1.4 Design should be kept simple [1].

5.1.5 The introduction of new designs or technologies aimed at improving operational/economic efficiency should not bring the current levels of safety beyond stated acceptability criteria [6].

5.1.6 **'Developmental' technologies should be considered for introduction into operational systems only after sufficient objective assurance has been gained to support their safe application.**

5.2 RATIONALE

5.2.1 Full validation of complex and or new designs is not always a feasible task.

5.2.2 Building reliable safe software is not a trivial task. Nor is demonstrating that safety and reliability have been attained.

6. PROCESS AND PRODUCT BASED SAFETY ASSURANCE

6.1 PRINCIPLE

6.1.1 **Safety assurance should be gained by appropriate management of the development process. This involves:**

(a) using system management and engineering practices established by relevant national/international standards to minimise and control systematic and random failures [4, 17]; and

(b) timely monitoring of the development process and its outcome through a set of appropriate metrics [15].

6.2 RATIONALE

6.2.1 Safety is most effectively achieved by building it into the system [5, 14].

6.2.2 Safety arguments should be substantiated with objective and traceable evidence [2, 5].

6.3 EXAMPLES

6.3.1 The following are examples of how this principle should be supported.

6.3.2 Planned software safety activities should integrate with system safety activities and should be documented either in a stand-alone plan or as part of the project plan [4, 5].

6.3.3 Reasoning for safety significant decisions should be recorded along with supporting evidence. It is essential that traceability of hazards associated with safety requirements and their resolution, is maintained at all levels (system, sub-system, component) [2, 12, 17].

6.3.4 Safety requirements should be explicitly stated (function and integrity), formally verified and monitored for continuing fulfilment and suitability [4].

6.3.5 Traceability should exist between safety requirements, design, implementation and ongoing operation [4, 14].

6.3.6 The configuration should be tightly controlled. This includes a well documented and formal change management process covering analysis, approvals, implementation and verification [4, 14].

6.3.7 Procedures should be adopted to minimise the introduction of systematic faults and applied in the framework of an organised development approach. These procedures should be available as a range of techniques applicable to different phases of development.

Guidance to the selection of techniques according to the criticality of the system should also be documented and used to ascertain a supplier's capability and/or included as part of the contractual specification [4]. Examples of some techniques applicable during different phases are:

- (a) Software Safety Requirements Specification: structured methods such as Data Flow Diagrams, formal specification notations such as Z;
- (b) Design & Development – Architecture Design: Fault Detection and Diagnosis, Diverse Programming, Dynamic Reconfiguration, etc.;
- (c) Design & Development – Development Tools and Programming Languages: Programming Language, Language Subset; Certified Translator or proven in use, CASE tools, etc.;
- (d) Design & Development – Detailed Design: Structured Methodology, Defensive Programming, Structured Programming, Analysable Programs, etc.;
- (e) Integration Testing: Functional and Black Box, performance, interface, etc.;
- (f) Modification: Impact Analysis, Re-verification of changed and affected modules, Revalidate Complete System;
- (g) Functional Safety Assessment: FTA, FMECA, Complexity Metrics.

6.3.8 The design should incorporate features for controlling both random and systematic failures during operations. A library of potential features should be available. Guidance for the use of those features should take into account the criticality of the system. (Note: IEC61508 recognises systematic faults of 4 different types: hardware design, environmental stresses, operator mistake, software design) [4].

- (a) Examples of such features are: Fault Detection and Diagnosis, Diverse Hardware, Graceful Degradation, Input Acknowledgement, Modification Protection, etc.

6.3.9 Software and systems engineering effort should be integrated and managed through policies and procedures which refer to, or are based on, nationally or internationally accepted standards applicable to all systems being procured.

6.3.10 Verification of safety requirements should be done progressively during development (part by the developer) and the degree of independence of the verification effort should be commensurate with the criticality of the system and task.

6.3.11 Independent audits should be carried out during development in order to monitor compliance with development methods and safety plans. The degree of independence should correspond with the required integrity level. The highest level should call for a completely independent organisation. Audit results should be used as input during safety assessments [4].

6.3.12 Software based systems normally have a large number of states rendering exhaustive testing impractical. This limited assurance should be recognised and balanced by seeking appropriate confidence in the design [5].

6.3.13 Negative and stress testing should always be included.

6.3.14 Policies should give guidance to both the handling of failures during, and the design of, tests that subject an already integrated and installed system to a realistic input

environment for a set period of time. Such a test is sometimes referred to as 'operational test'. Examples of topics which should be dealt with by policies are:

- (a) The operational profile used during testing should represent a comprehensive and realistic operating environment. Documentation should substantiate this.
- (b) Criteria should be established to define what failures render a system non-commissionable. They should describe how frequency of failure and severity of consequence are taken into account.
- (c) Guidance should be given to determine the length of time an 'operational test' is to be run without failures, before a system can be considered reliable enough to be commissioned. The guidance should be supported by a documented rationale.
- (d) Should the operational test demonstrate that the product delivered by the development process is actually less reliable than expected, doubts will be cast over the actual effectiveness of the process. Therefore, policies should specify and justify whether, after each fault correction, the new and supposedly correct software is to be subject to the previous operational test or one that is actually more stringent [9].

For example, it would be expected that after strictly following an appropriately rigorous development process, no high-consequence failure occurs at all during an appropriately designed and long operational test. Should such a failure occur, it would be sensible to subject the new software to a more 'thorough' operational test than before.

- 6.3.15 The decision to release software for operational use should be based on the level of assurance that an appropriate design process has factually been followed as well as the actual results from a reliability growth program. This assurance should be sought irrespective of the type or architecture of the hardware-host (e.g. mini computer, personal computer, micro computer, programmable logic controller, networked, standalone, etc.) and regardless of the software maturity (e.g. already proven in use, modification of existing-proven software, a newly released commercial off-the-shelf product or a totally new development). Note that depending on the level of software maturity as well as visibility granted to the service provider, the actual evidence may take different forms [15].
- 6.3.16 Suppliers should be required to 'certify' that the delivered product, the management of its development and the development method comply with the service provider requirements and specified national/international standards.
- 6.3.17 Controls should be exercised before contract signature in order to evaluate subcontractors (suppliers to the service provider) and explicit evaluation criteria should be specified. For example:
 - (a) demonstrated degree of expertise in the field;
 - (b) demonstrated degree of process maturity as an organisation;
 - (c) current commitments;
 - (d) certification against quality assurance standards;
 - (e) use of national/international system and software safety standards.

References

1. Brooks, F. P. 1987. No silver bullet. *Computer*, April, 1987.
 2. Civil Aviation Safety Authority. 1996. Safety Regulation of Airservices, Safety Regulatory Arrangements and Standards.
 3. Dale, C. 1990. *Software Reliability Issues*. Software Reliability Handbook, ed. Paul Rook, Elsevier Applied Science.
 4. International Electro-technical Commission (IEC) . 1995. *IEC61508 draft; Functional Safety: Safety Related Systems*.
 5. Leveson, N. 1995. *Safeware, System Safety and Computers* Addison Wesley.
 6. Littlewood, B and Stringini L. 1992. *The Risks of Software*. Scientific American, November.
 7. Littlewood, B. 1990. *Modelling Growth in Software Reliability*. Software Reliability Handbook, ed. Paul Rook, Elsevier Applied Science.
 8. Littlewood, B. 1991. *Limits to evaluation of software dependability*. Proceedings: Software Reliability and Metrics Conference. Centre for Software Reliability, UK. Elsevier Applied Science.
 9. Littlewood, B and Wright, D. 1997. *Some Conservative Stopping Rules for the Operational Testing of Safety Critical Systems*. IEEE Transactions on Software Engineering vol23, No11, pp673-683, 1997.
 10. Perow, Ch. 1984. *Normal Accidents: Living with High Risk Technologies*. Basic Books.
 11. Radio Technical Commission for Aeronautics (RTCA). 1992. *DO178B: Software Considerations in Airborne Systems and Equipment Certification*.
 12. Society of Automotive Engineers (SAE). 1996. *ARP 4754 Certification Considerations for Highly Integrated Systems*.
 13. Society of Automotive Engineers (SAE). 1996. *ARP 4761. Guidelines and Methods for conducting the safety assessment process on civil airborne systems and equipment*.
 14. Stephenson, J. 1991. *System Safety 2000*. Van Nostrand.
 15. United Kingdom National Air Traffic Services (UK NATS) . 1996. *Safety Management Manual*.
 16. Underwood, A. 1996. *Computers and Safety - An overview*. Australian Computing Society Technical Committee on Safety Critical Systems Seminar.
 17. United States of America Department of Defence (USA DOD). *MILSTD-882C/1993: System Safety Program Requirements*.
-