Australian Government
Civil Aviation Safety Authority

Advisory Circular

AC 171-04(0) NOVEMBER 2006

SOFTWARE AND ITS USE IN AERONAUTICAL TELECOMMUNICATION AND RADIO NAVIGATION SERVICES

CONTENTS

1. REFERENCES

- Civil Aviation Safety Regulations (CASR) Part 171 – Aeronautical Telecommunication and Radio Navigation Service Providers
- Manual of Standards - CASR Part 171
- Refer to the Annex to this AC for text references.

2. PURPOSE

This Advisory Circular (AC) provides guidance and information on software system acquisition, development, implementation and support to organisations that are certified, or seeking certification as an aeronautical telecommunication and/or radio navigation service provider under the provisions of CASR Part 171.

3. STATUS OF THIS AC

This AC is essentially a rewrite (with some minor editorial changes for conversion to an AC) of CAAP CASA/AA MOU. AIRWAYS-2(0) that was issued in July 1999. That CAAP has been replaced by this AC.

Advisory Circulars are intended to provide advice and guidance to illustrate a means, but not necessarily the only means, of complying with the Regulations, or to explain certain regulatory requirements by providing informative, interpretative and explanatory material.

Where an AC is referred to in a 'Note' below the regulation, the AC remains as guidance material.

ACs should always be read in conjunction with the referenced regulations.

4. DEFINITIONS

4.1 In this AC, the definitions of the following terms are the same as those given in CASR Part 171:

Aeronautical telecommunication service: Refer to the definition at regulation 171.012 'Meaning of *telecommunication service'*.

Radionavigation service: Refer to the definition at regulation 171.010 'Interpretation'.

Where the word *'service'* or *'system'* is used, they are synonymous.

5. GUIDING PRINCIPLES FOR USE OF SOFTWARE IN AERONAUTICAL TELECOMMUNICATION AND RADIONAVIGATION SYSTEMS

5.1 Aeronautical telecommunication and radionavigation systems being deployed today are not only increasingly complex and integrated, but also increasingly software based. It has therefore become imperative to consider safety aspects by a structured approach commencing early in the life-cycle of a new system, preferably at the operational concept phase.

5.2 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.).

5.3 The guidelines in this AC have been developed to outline those common principles, as CASA has adopted a non-prescriptive approach to regulating aeronautical telecommunication and radionavigation service providers.

5.4 The principles apply equally across different application domains irrespective of the specific project organisation, engineering or management standards and practices applied.

5.5 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.

5.6 The principles are:

(a) Integrated System Safety;

(b) Specified Integrity Levels;

(c) Conservative System Architecture and Technologies;

(d) Process and Product based Safety Assurance.

Note: Inasmuch as these guidelines relate to software in the context of system and safety issues, the subjects covered may well apply to other domains.

6. INTEGRATED SYSTEM SAFETY

6.1 Principle

6.1.1 Taking a holistic approach, system safety should integrate and cut across all disciplines within a controlled and planned development environment and through all lifecycle phases. There should be no stand alone and unconnected safety activity, including software technology [see references 5, 12, 15, 17 at the Annex to this AC].

6.1.2 Agencies or individuals implementing aeronautical telecommunication and/or radionavigation services 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 [see reference 5].

6.2 Rationale

6.2.1 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 [see references 5, 10].

6.2.2 System safety is about preventing accidents by building-in safety through the application of management and engineering techniques during all lifecycle phases [see reference 14].

6.2.3 System safety provides the highest benefits when applied from early lifecycle phases (i.e. operational concept, early design) [see references 5, 14].

6.2.4 The issue of whether the requirements themselves are safe is as important as whether the implemented service satisfies the requirements [see reference 5].

7. SPECIFIED INTEGRITY LEVELS

7.1 Principle

7.1.1 Software safety assurance should be based either on the concept of specified Systematic Safety Integrity Levels [see reference 4] or Software Levels [see reference 11]. 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 [see references 4, 11, 15].

7.2 Rationale

7.2.1 Functions of differing criticality are implemented in software.

8. CONSERVATIVE SYSTEM ARCHITECTURE AND TECHNOLOGIES

8.1 Principle

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

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

8.1.3 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 [see reference 8].

8.1.4 Design should be kept simple [see reference 1].

8.1.5 The introduction of new designs or technologies aimed at improving operational/economic efficiency should not result in a lowering of the current levels of safety beyond the level of the stated acceptability criteria for the service [see reference 6].

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

8.2 Rationale

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

8.2.2 Building reliable safe software is not a trivial task; nor is demonstrating that safety and reliability have been attained.

9. PROCESS AND PRODUCT BASED SAFETY ASSURANCE

9.1 Principle

9.1.1 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 [see references 4, 17]; and

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

9.2 Rationale

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

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

9.3 Examples

9.3.1 The following are examples of how this principle should be supported:

(a) 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 [see references 4, 5].

(b) 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) [see reference 2, 12, 17].

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

(d) Traceability should exist between safety requirements, design, implementation and ongoing operation [see references 4, 14].

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

(f) 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 [see reference 4]. Examples of some techniques applicable during different phases are:

(i) Software Safety Requirements Specification: structured methods such as Data Flow Diagrams, formal specification notations such as Z;

(ii) Design & Development – Architecture Design: Fault Detection and Diagnosis, Diverse Programming, Dynamic Reconfiguration;

(iii) Design & Development – Development Tools and Programming Languages: Programming Language, Language Subset; Certified Translator or proven in use, CASE tools;

(iv) Design & Development – Detailed Design: Structured Methodology, Defensive Programming, Structured Programming, Analysable Programs;

(v) Integration Testing: Functional and Black Box, performance, interface;

(vi) Modification: Impact Analysis, Re-verification of changed and affected modules, Revalidate Complete System;

(vii) Functional Safety Assessment: FTA, FMECA, Complexity Metrics.

(g) 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 four different types: hardware design; environmental stresses; operator mistake; software design) [see reference 4]. Examples of such features are: Fault Detection and Diagnosis; Diverse Hardware; Graceful Degradation; Input Acknowledgement; Modification Protection.

(h) 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.

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

(j) 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 [see reference 4].

(k) 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 [see reference 5].

(l) Negative and stress testing should always be included.

(m) Policies should give guidance to both the handling of failures during, and in 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:

(i) The operational profile used during testing should represent a comprehensive and realistic operating environment. Documentation should substantiate this.

(ii) 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.

(iii) 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.

(iv) 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 [see reference 9].

(v) 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.

(n) 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 [see reference 15].

(o) 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.

(p) 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:

(i) demonstrated degree of expertise in the field;

(ii) demonstrated degree of process maturity as an organisation;

(iii) current commitments;

(iv) certification against quality assurance standards; and

(v) use of national/international system and software safety standards.

Patrick Murray
Group General Manager
Air Transport Operations Group

ANNEX

REFERENCES

1. Brooks, F. P. 1987. No silver bullet. Computer, April, 1987.
2. Civil Aviation Safety Regulations (1998) – CASR Part 171 – Aeronautical Telecommunication and Radionavigation Service Providers
3. Dale, C. 1990. *Software Reliability Issues*. Software Reliability Handbook, ed. Paul Rook, Elsevier Applied Science.
4. International Electro-technical Commission (IEC) 61508, Functional Safety of Electrical/Electronic/Programmable Electronic 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.
18. Radio Technical Commission for Aeronautics (RTCA). 2002. DO-278: *Guidelines for Communication, Navigation, Surveillance and Air Traffic Management (CNS/ATM Systems Software Integrity Assurance.*