

# THE CHALLENGES AND ADVANCES IN COTS SOFTWARE FOR AVIONICS SYSTEMS

Eur Ing P.J. Parkinson

Wind River, United Kingdom. Email: Paul.Parkinson@windriver.com

**Keywords:** avionics certification COTS safety security.

## Abstract

This paper discusses the challenges involved in the development, deployment and through-life support of civil and military avionics applications. The challenges and pressures which have led to the increasing widespread adoption of COTS software components for mission-critical and safety-critical avionics in relation to DO-178B / ED-12B and UK MOD Defence Standard 00-56 will be considered. The potential pitfalls and benefits of COTS adoption will also be considered, with reference to software portability, risk reduction and planned obsolescence. The role open standards in Integrated Modular Avionics (IMA) and the drive towards modular and incremental safety certification will also be considered.

## 1 Introduction

In recent years, there has been a marked increase in the use of commercial off-the-shelf (COTS) software in civil and military avionics systems for mission-critical and safety-critical systems. There have been a number of distinct contributory factors which when combined have accelerated the adoption of COTS. We will consider these drivers for change and the effects of their combined input in the following sections, and future challenges which may be addressed using COTS software.

## 2 Drivers for Change

Historically, avionics systems have utilised a federated architecture where an individual avionics Line Replaceable Unit (LRU) has performed a specific dedicated function, such as an air data computer. This architecture is inherently robust, because a hardware or software failure on this LRU may not necessarily impact other avionics functions. However, this architecture can be very costly in terms of space, weight and power (SWaP), and also the cabling requirements between LRUs.

In addition, the software content in avionics systems has increased dramatically in recent years, for example, a military fast jet in service around 1990 would have had several hundred thousand source lines of code (SLOC), whereas the F-35 Lightning II will have around seven million SLOC in the airborne systems, and a further 7m SLOC in test systems.

This software complexity is being driven by requirements for increased application functionality, and this has placed additional workload on the LRU computing platforms, sometimes requiring higher performance variants to be used. However, these rising performance demands have also faced the problem, particularly in military but also civil avionics, of the decline in the availability of military grade components as a result of the US Defense Secretary Perry's memorandum [12]. It is unlikely that initial processor selections would be able to support the current and future capability requirements of individual systems for the whole of the in-service life of the platform. This is especially true in the case of military avionics systems where the in-service lifetimes of some military aircraft are being increased due to defence budget pressures (e.g. the B-52 bomber is expected to remain in service until 2040, with an in-service lifetime of 95 years [3]).

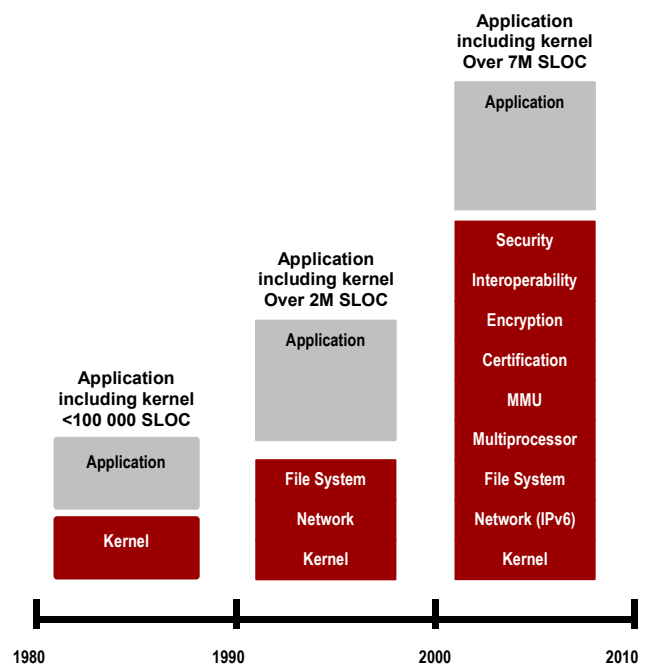


Figure 1: Software complexity

The civil avionics domain has faced slightly different pressures, as airframe manufacturers have sought to achieve greater interoperability between LRUs in order to reduce their dependency on individual Tier 1 suppliers (systems

integrators), resulting in increased supplier competition and reduced programme costs.

It is therefore interesting to note that the adoption of an Integrated Modular Avionics (IMA) architecture can be used to address these issues in both civil and military avionics systems. The IMA approach (sometimes also known as *Integrated Modular Systems*) advocates the use of a common shared computing platform to host multiple applications. This in addresses the issue of Space, Weight and Power (SWaP) by reducing processor count and increasing processing density. The reduction in processor types can contribute to reducing the burden of processor obsolescence in avionics systems; this can be significant, as some military aircraft use over eighty processor variants.

However, to derive real benefit from the shared computing platform, and reduce hardware dependency on the underlying processor architecture, software abstraction should be used wherever possible. The simplest level of abstraction is to use of a appropriate High-Level Language (HLL) such as Ada 2005 [1] or ANSI 'C' rather than processor-specific assembly language, this would make the migration from an obsolescent Motorola 68000 processor to a current Freescale or IBM PowerPC™ architecture more straightforward. This processor dependency can be further reduced when combined with the use of a portable software abstraction layer, and this is usually implemented as a library of services with a published application programming interface (API).

### 3 The Benefits and Pitfalls of COTS

Integrated Modular Avionics architectures usually define an OS or Application Executive (APEX) layer to abstract from the underlying hardware, and also to implement and enforce temporal and spatial partitioning between the multiple applications on the platform to ensure that they do not interfere with each other. Although some avionics suppliers have implemented their own real-time operating system (RTOS) or executives to support the development of avionics applications and some have even made them available as commercial products, other suppliers have often been reluctant to use these proprietary implementations for fear that it might potentially put them at a commercial disadvantage in competitive situations, and so have tended to adopt COTS implementations instead.

When selecting a COTS implementation, it is important to ensure that both the development tools framework and the OS runtime provide open standards-based interfaces to provide framework extensibility and also application portability, rather than using unpublished closed or proprietary interfaces; otherwise the hardware abstraction gained could be compromised by vendor lock-in. This is sometimes referred to as the “COTS Open vs COTS Closed” problem. In a “COTS Open” system, all of the interfaces (see figure 2) from the application API through systems configuration to toolset and target integration are implemented using published interfaces based on open standards (e.g. ARINC

653 [2] and POSIX PSE52 [10]), which enables the development of truly portable applications. In a “COTS closed” system, proprietary APIs and build systems are used which result in vendor lock-in or interfaces which are not published at all, which limits toolset integration with other vendors and locks in this vendor’s tools.

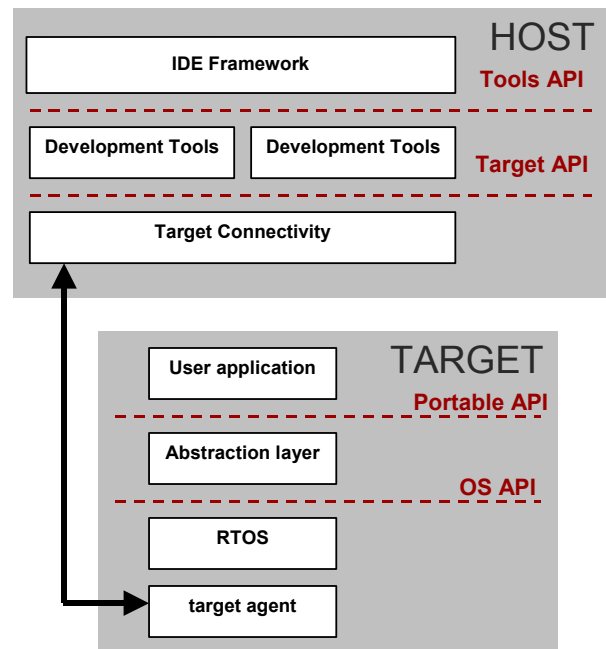


Figure 2: Tools & platform architecture

There are of course many industry standards and de facto standards in the software industry, but it is not always clear which of these standards are truly open and portable, and which are vendor-dependent or vendor-driven – which could provide a dead-end with limited portability.

For example, when considering the use of a programming language subset for the development of a safety-critical application, care should be taken to ensure that:

- (i) The subset definition is not proprietary to a single vendor, but there is an industry organisation or open standards body;
- (ii) There is support for the language subset from multiple COTS vendors to prevent vendor lock-in, but also to de-risk long term programme support by providing multiple options for future migration.

In this respect, the Ada 2005 Ravenscar Profile and MISRA-C [5] are positive examples, and in the future a MISRA C++ standard may also provide a suitable language subset.

This standardisation approach, providing choice and flexibility based on open standards, is one of the tenets of the Device Software Optimisation (DSO) philosophy, which has

been advocated by Wind River in recent years and is gaining widespread industry adoption.

#### 4 COTS Software Certification

We have already considered a number of issues in the selection and use of COTS software in avionics systems, but one of the most important issues is safety certification for the following reasons:

- (i) The cost of safety certification of software systems, especially at higher levels of criticality, can be far greater than the development costs;
- (ii) The software must be fit for purpose, as a poorly designed or developed system may never be suitable for certification;
- (iii) There is a risk that a certification approach may lead to a proprietary or unsupportable system, undoing any good work in the development phase;
- (iv) Systems are likely to be enhanced through their in-service lifetimes through mid-life updates and technology insertions, and should support incremental certification.

These issues will be explored in turn in the following subsections.

##### 4.1 Certification Standards and Costs

There are a number of safety certification standards which apply to UK avionics systems in a variety of contexts. In June 2007, the UK Ministry of Defence published Defence Standard 00-56 Issue 4 [11] which clarified a number of issues in the earlier Issue 3 interim standard. The revised standard adopts a goal-based philosophy, and is much less prescriptive about the approach to be taken than the earlier Issue 2 standard. This enables the use of processes from other relevant safety standards and the possibility of reuse certification evidence developed for other safety standards - provided that it can be demonstrated that the software has been developed to the appropriate level of safety. This provides the prospect of software which has been developed under RTCA DO-178B [8] and the EUROCAE ED-12B [6] for civil programmes certified by the FAA in the US and EASA in Europe respectively. In recent years, the US military has adopted DO-178B, although US military programmes are not certified by the US Federal Aviation Administration (FAA).

DO-178B and ED-12B defines a number of software levels for different systems and the effects of failure. These are illustrated in table 1. DO-178B defines a number of processes to be undertaken across the software lifecycle, and the processes place an emphasis on requirements traceability, so that each source line of code (SLOC) must be derived from a

requirement, and any code not mapped to a requirement is regarded as dead code.

Failure Condition	Software Level	Outcome
Catastrophic	Level A	Death or injury
Hazardous / Severe - Major	Level B	Injury
Major	Level C	Unsafe workload
Minor	Level D	Increased workload
No Effect	Level E	None

Table 1: DO-178B/ED-12B Software Levels

DO-178B also defines specific objectives at each software level (66 for Level A, 65 for Level B, 57 for Level C and 28 for Level D), and some of the additional objectives can have a significant impact on the testing effort required. For example, the code coverage testing requirement introduced at Level C requires statement level coverage to be performed. Level B also requires decision coverage, and Level A requires Modified Decision / Condition Coverage (MC/DC), which involves testing complex decision conditions which result in each combination of conditions being satisfied at least once.

These testing and code coverage requirements are likely to be more rigorous than those undertaken for the development of general-purpose commercial grade software. This can create a hurdle for the use of previously developed software and new COTS software developments in avionics systems. For example, consider the MCDC testing requirement for the following Ada code fragment:

```
if A=0 and then B<2 and then C>5 then
P; end if;
```

This contains three variables, three conditions and four MCDC cases (as shown in Table 2). For DO-178B Level B certification, two test cases which result in both the execution and non-execution of statement P are required, whereas for Level A certification, all four possible test cases need to be generated.

A=0	B<2	C>5	P
T	T	T	T
F	?	?	F
T	F	?	F
T	T	F	F

Table 2: MCDC Test Cases

##### 4.2 Fitness for Purpose

For the most critical applications, which we could call “*safety critical*”, DO-178B Level A, Level B and possibly Level C ordinary general purpose COTS software implementations are not appropriate; but specifically developed safety-critical implications which have been developed in accordance with the software safety standards must be used. (This is also in

accordance with UK MOD guidance on the use of COTS software in military avionics systems [4]).

However, at the lower levels of safety-criticality, there also appears to be a growing trend to use an evidence-based approach to certification as an alternative to following safety-critical software development guidelines. An evidence-based approach usually involves three aspects: commercial software development process information, in-service evidence, and defect tracking/correction data. This approach may be acceptable to certification authority at DO-178B Level D and Level E or Def Stan 00-56 SIL 1, where ordinary commercial grade software components may be accepted with additional supporting information. This approach does not address the testing and code coverage requirements at higher software levels of DO-178B & ED-12B, and also there are limits to the relevance of in-service data as the configuration of a software component and underlying hardware architecture (e.g. processor variant) could differ significantly from the system currently under development.

The use of COTS software should also be considered within the context of the overall system, in particular in relation to whether other software modules or applications have different levels of safety-criticality. Systems which have multiple-levels of criticality (sometimes referred to as *mixed-criticality*) can be supported by Integrated Modular Avionics platforms (as shown in figure 3) provided that the RTOS implements and enforces temporal and spatial partitioning. This is a fundamental requirement in order to ensure that individual applications, possibly of differing levels of safety criticality, cannot interfere with each other. The RTOS must be safety-certified to the level of the most critical application. The rationale for this is that each layer of software, from hardware-dependent code upwards, provides a foundation for the layer above it because the higher-level layers use the services (via APIs) provided by the layer below. Therefore, to provide a robust and dependable system, the lower layers must be safety-certified to the level of the most critical application or layer above.

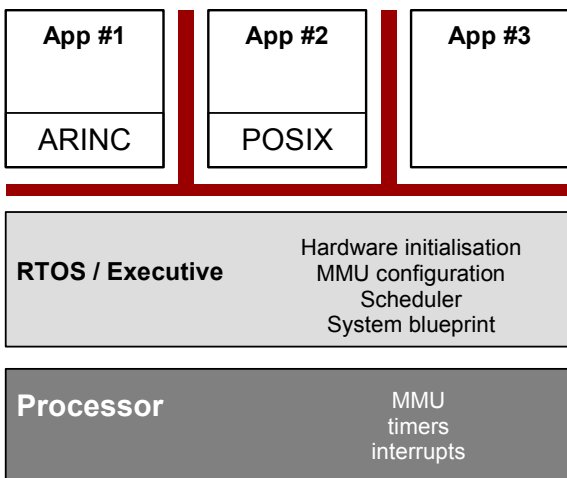


Figure 3: Integrated Modular Avionics architecture

It has been suggested that this foundation approach can be bypassed in order to implement a federated architecture with mixed-criticality, by writing a ‘high-integrity’ code block for a higher safety-critical level which executes with interrupts locked, on top of lower-level software layers or RTOS developed to a lower-level of safety-criticality. However, this approach is fundamentally flawed because it makes invalid assumptions about the state of the system when it executes, as hardware initialisation has been performed by lower layers. For example, on a PowerPC processor, the lower-level layers will have set up the processor memory map, including chip select registers, and initialised the memory management unit (MMU), system clock, timers, etc. Unless the ‘high-integrity’ code block performs all of the hardware and OS initialisation itself, then it is making invalid assumptions about the state of the system. The solution therefore to use an RTOS which supports the same level of criticality.

Special consideration is required for safety-certification under UK MOD Defence Standard 00-56 at software integrity level (SIL) 4, as this involves the use of formal methods which can result in significant costs. Until now, COTS software has not addressed SIL 4 systems, but this could change in the near future with the introduction of secure OS platforms which may use formal methods during development to achieve security certification.

### 4.3 Reusable and portable safety evidence

It is important to ensure that the RTOS safety certification evidence is readily available as a true COTS product, according to the DO-178B [8] definition of COTS:

*Commercial off-the-shelf (COTS) software - Commercially available applications sold by vendors through public catalog listings. COTS software is not intended to be customized or enhanced. Contract-negotiated software developed for a specific application is not COTS software.*

This is because if the certification evidence is restricted or tied to a particular programme, it could prevent it from being used on other programmes. This represents a significant risk with service-based certification, where the certification evidence for the RTOS is generated along with the application during the certification efforts for an overall system, and the RTOS certification evidence cannot be decoupled from the application, making it difficult to reuse.

### 4.4 Modular & Incremental Certification

In many avionics systems, there is often a need to perform a mid-life update or technology insertion during the in-service lifetime to provide additional functionality or capability. This can be problematic, as the modification of an already certified system will require re-certification. In particular, research has shown that “the cost of change is proportional to the size of the system, not the size of the change” [7]. In a large

federated system, this can have a disastrous effect on the incremental certification cost for an update to the system.

The Integrated Modular Avionics model advocates the use of modular system components, however this alone does not provide a guarantee of modular and incremental certification. For example, if the IMA implementation combines the binary modules for individual application partitions and RTOS into a single monolithic image, then this could not support incremental certification.

However, it has been shown that an IMA architecture using an ARINC 653 [2] implementation with role-based configuration under RTCA/DO-297 [9] can produce a platform suitable for modular and incremental certification. This approach uses XML-based configuration to define the system and partition attributes, and produces a payload stream of binary modules for the individual applications, OS and system configuration file (a.k.a. *system blueprint*).

## 5 Challenges for the Future

In the future, initiatives such as Network Enabled Capability (NEC) and Network Centric Warfare (NCW) will continue to drive increased connectivity between airborne military platforms in order to share information between coalition forces. These systems already implement communications security (ComSec) to ensure that information is securely transferred between nodes without eavesdropping, but it is anticipated that there will be a growing need to implement information security (InfoSec) measures to ensure that the appropriate information flows of different classifications of data (Top Secret, Secret, Unclassified). In addition, as the capabilities of civil air traffic management increase to enable Free Flight, there is a growing requirement for military systems to be able to communicate with civil systems.

The impact of these information security requirements on the platforms operating systems, is that they will need to be able to be multi-level secure (MLS) and enforce authorised data flows whilst preventing the unauthorised disclosure of sensitive information. These requirements are being addressed through COTS RTOS implementations based on the Multiple Independent Levels of Security (MILS) architecture [13], and it is anticipated that they could provide the foundation for a safe and secure platform.

## About the Author

Eur Ing Paul Parkinson BSc(Hons) CEng MBCS MIET is a Senior Systems Architect with Wind River, working with customers in the Aerospace & Defence sectors in the UK and Nordic countries. Paul's professional interests include Integrated Modular Avionics (IMA) and Intelligence Surveillance Target Acquisition Reconnaissance (ISTAR) systems. Paul blogs on A&D industry issues on the Wind River website at URL <http://blogs.windriver.com/parkinson>.

## References

- [1] Ada 2005 Language Reference Manual. URL <http://www.adaic.com/standards/ada05.htm>
- [2] "Avionics Application Software Standard Interface" ARINC 653-1 (2003). URL <http://www.arinc.com>
- [3] B-52 Bomber, US Air Force Link website. URL <http://www.af.mil/factsheets/factsheet.asp?fsID=83>
- [4] B. Dobbing, "Practical Guide to Certification and Re-Certification of AAvA Software Elements - COTS Real-Time Operating Systems", S.P1211.40.2, UK Ministry of Defence (2004).
- [5] "Guidelines for the use of the C language in critical systems", MISRA-C:2004, MISRA. URL <http://www.misra-c2.com>
- [6] "Software Considerations in Airborne Systems and Equipment Certification", ED12B, EUROCAE (1992). URL <http://www.eurocae.org>
- [7] G. Holton, "BAE SYSTEMS Military Air Solutions", UK MOD Military Avionics Technology Exhibition (2007).
- [8] "Software Considerations in Airborne Systems and Equipment Certification", DO-178B, RTCA (1992). URL <http://www.rtca.org>
- [9] "Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations", DO-297, RTCA (2005). URL <http://www.rtca.org>
- [10] "POSIX 1003.13-2003 PSE52 Real-Time Controller Profile", The Open Group (2003). URL <http://get.posixcertified.ieee.org/docs/pse52-2003.html>
- [11] "Safety Management Requirements for Defence Systems", Defence Standard 00-56 Issue 4, UK Ministry of Defence (2007).
- [12] W. Perry, US Secretary of State for Defense, "Specifications & Standards – A New Way of Doing Business", US DOD Memorandum (29 June 1994).
- [13] M. Vanfleet, NSA, "Multiple Independent Levels of Security", Open Group Security Forum (2002).