Software Defined Combat Systems

Best Practices for Developing Safe and Secure Modular Software

The Open Group FACE® and SOSA® Consortia 2025 Technical Interchange Meeting (TIM), by:

Alex Lim, LDRA Technology, Inc

Stephen Di Camillo, LDRA Technology, Inc

October 2025

Table of Contents

Executive Summary

Designing systems which are highly adaptable and interoperable is particularly important in industries like aerospace and defense because of the need to incorporate rapidly evolving technology into platforms with a long service life. As a result, aerospace and defense organizations are embracing – and in some cases mandating - a Modular Open Systems Approach (MOSA). A MOSA emphasizes modularity, interoperability, and the use of open standards, fostering an environment where components from diverse sources can work together seamlessly. Open architecture standards such as The Open Group FACE® (Future Airborne Capability Environment®) Technical Standard, Aeronautical Radio, Inc (ARINC) Specification 653, and RTCA/DO-297 promote modularity, reusability, and interoperability in software systems. These standards enable developers to create adaptable components that can be integrated across multiple platforms, reducing duplication and fostering flexibility.

While a MOSA offers significant benefits, it also presents unique challenges. Safety and security are especially important given the requirements for component modularity and portability, where components may be widely re-used across many systems. Given the rapid pace of technological innovation and understanding that most of this innovation is realized through software, it's imperative that MOSA-aligned software components comply with appropriate functional safety and security standards.

In this paper, we'll explore some of the challenges of following a MOSA for developing safe and secure modular software components. We'll also discuss several best practices for software development and verification that support a MOSA and facilitate compliance with appropriate functional safety and security standards.

Modular Design

Overview

The basic concept of modular design is not new – in fact, the concept of modular design as applied to construction dates back thousands of years. The Egyptians used prefabricated components to construct the pyramids, with massive stone blocks carefully cut and transported to the construction site to be assembled. Similarly, the Romans employed modular techniques in constructing aqueducts and roads – some of these are still in use today [WM1].

Over time, the use of modular design has advanced beyond construction, and is used to develop everything from toys and televisions to automobiles, commercial and military aircraft, and cruise ships weighing nearly 250,000 tons and carrying 7,600 passengers [IS1].

Simply stated, modular design is an approach where a product or system is divided into smaller, independent, and interchangeable units called modules. These modules can be designed, built, and tested separately, then integrated to create a larger system. As systems have grown in complexity while the market demands for access to new technology drive manufacturers to continue to strive to shorten time-to-market, the economics of developing systems from scratch have largely gone by the wayside. Modular design is an essential strategy to reduce the cost and effort of bringing new products to market.

The main advantages of modular design include design flexibility, improved augmentation, and cost reduction. Due to grouping the components to each module, the designer can easily modify each module instead of changing the whole design. In addition, the system can be upgraded by adding new functions simply by plugging in a new module so that the system can be augmented within a specific range. Modular design relies on the concepts of product architecture and product platform concepts. Product architecture defines the system's components and their interfaces. A product platform is the foundation of common system components upon which product specific components can be interchanged to provide product variation [MD1].

This paper focuses on the application of modular design to the development, verification, and certification of software. In the next sections of the paper, we will briefly review both civilian and defense standards that embrace the concept of modular software.

Civilian Standards Supporting Modular Design

Several civilian standards include guidance around the development, verification, and certification of modular software:

RTCA/DO-297 and FAA AC 20-170

RTCA/DO-297 is formally known as "Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations." IMA is a shared set of flexible, reusable, and interoperable hardware and software resources that, when integrated, form a platform that provides services, designed and verified to a defined set of requirements, to host applications performing aircraft functions. Along with ARINC 653 and DO-248, DO-297 provides guidance for developing and certifying integrated modular avionics systems in civil aircraft [DO297].

Federal Aviation Administration (FAA) Advisory Circular (AC) 20-170 is formally known as "Integrated Modular Avionics Development. Verification, Integration and Approval using RTCA/DO-297 and Technical Standard Order C153." Together with DO-297, this advisory circular provides an acceptable means of compliance for the development, integration, verification, and installation approval of IMA systems. The guidance focuses on the unique aspects of IMA design assurance, particularly when hosting multiple software functions of varying safety-criticality on a single computing platform [AC170].

RTCA/DO-178C and FAA AC 20-148

RTCA/DO-178C is formally known as "Software Considerations in Airborne Systems and Equipment Certification." This standard provides recommendations for the production of software for airborne systems and equipment, focusing on assuring software performs its intended function with a level of confidence in safety. While the guidance in DO-178C applies to all software development, including the development of reusable software components, it falls short of providing guidance of benefit to the development of reusable software components [DO178].

FAA AC 20-148 is formally known as ""Reusable Software Components." This advisory circular includes guidance for the development, integration, and certification of reusable software components (RSCs) in airborne systems. Within the context of RTCA/DO-178C, this document outlines a means of compliance for developers seeking FAA acceptance and credit for reusing software components in future projects. This circular emphasizes the benefits of software reuse, including reduced development time, lower costs, and improved software quality [AC148].

ARINC 653

ARINC 653 (Avionics Application Software Standard Interface) is a specification that defines a standard interface between avionics application software and a Real-Time Operating System for IMA systems. The specification focuses on space and time partitioning in safety-critical avionics, allowing multiple applications with different criticality levels to run on the same hardware. Partitioning ensures that one application's failure does not impact others, enhancing system reliability and facilitating certification. ARINC 653 promotes modularity, facilitates upgrades and maintenance, and reduces the cost of certification by allowing for independent testing and validation of individual partitions [A653].

POSIX

POSIX (Portable Operating System Interface) is a family of standards specified by the Institute of Electrical and Electronics Engineers (IEEE) and The Open Group for maintaining compatibility between operating systems. The standards define a common set of rules and specifications for operating systems, allowing software to run on different operating systems without requiring major modifications. By conforming to POSIX, operating systems and software developers can achieve greater interoperability, reducing the effort required to port applications between different systems [POSX].

Defense Standards Supporting Modular Design

Similar to the commercial world, there are several defense standards include guidance around the development, verification, and certification of modular software:

Best Practices for Developing Safe and Secure Modular Software

MOSA

The Modular Open Systems Approach (MOSA) is a strategic framework that emphasizes designing and developing complex systems in a modular, interoperable, and flexible manner. While MOSA is not a formal standard, it does describe principles, goals, and objectives that form the framework for systems acquisition and development. The MOSA approach is widely adopted across various industries, particularly where technological adaptability and long-term lifecycle management are crucial. Key principles and objectives of MOSA include employing modular design, improving interoperability, using open standards, and enabling vendor independence. MOSA promotes the design of systems as a collection of discrete, interchangeable components. This modular structure allows for easier upgrades, maintenance, and scalability. By isolating system functions within modules, organizations can update or replace parts of the system without impacting the entire architecture, reducing downtime and cost [MOSA].

FACE® Technical Standard

The FACE Technical Standard provides a standardized approach to software architecture for airborne systems, facilitating interoperability across different platforms and reducing the risk of vendor lock-in. By adhering to the FACE Approach, developers ensure that their software components can be easily integrated with other FACE Conformant software components, promoting modularity and flexibility. FACE Conformance not only simplifies integration but also simplifies future upgrades and expansions.

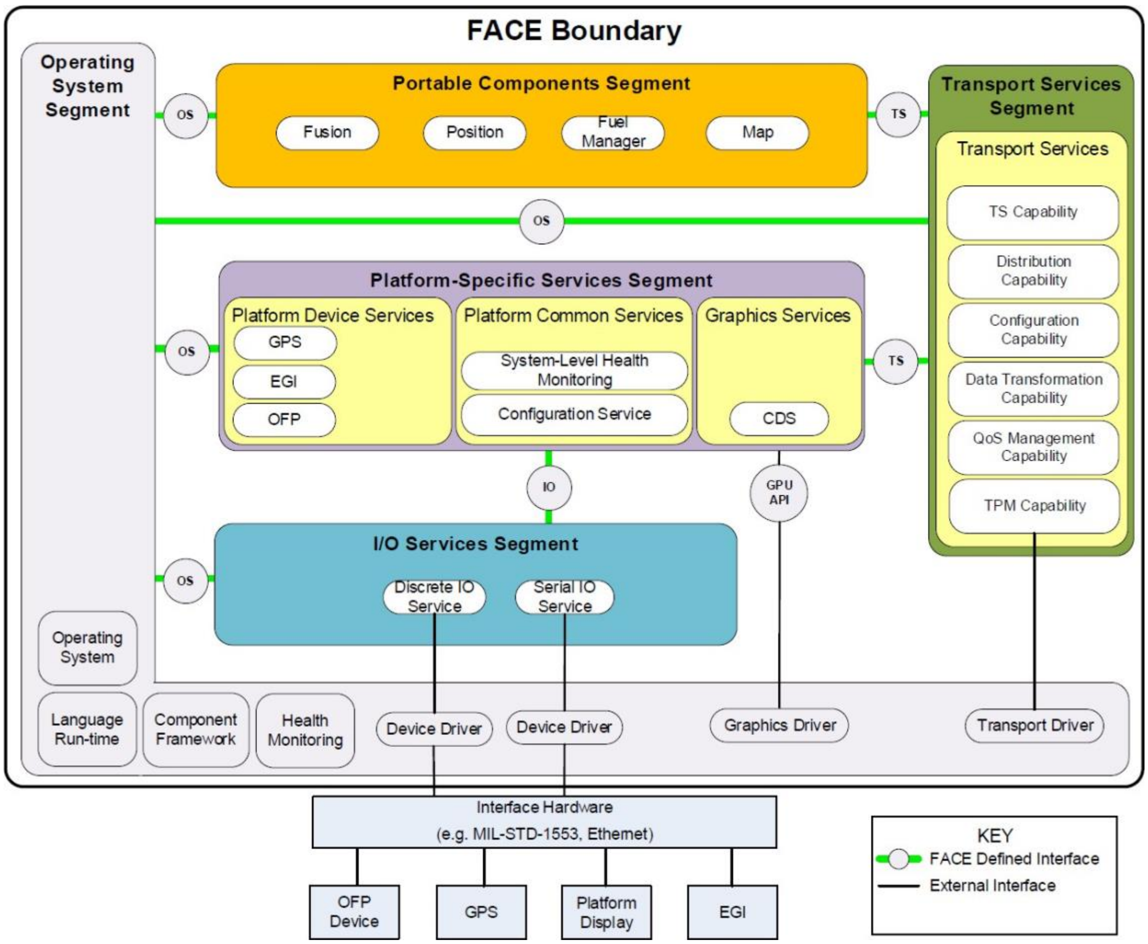

Figure 1 - FACE Reference architecture

The FACE Reference Architecture defines distinct segments, each specifying how components within and across these segments should communicate with each other (**Error! Reference source not found.**). This structured approach ensures consistent and reliable interactions among various system components, further facilitating interoperability and integration within the FACE ecosystem [C232].

PYRAMID

PYRAMID aims to make legacy and future air mission systems affordable, capable, and adaptable by adoption of an open systems architecture approach and systematic software re-use. A key goal of the PYRAMID program is to maximize adoption on both a national and international level. To achieve this, the PYRAMID Technical Standard and Guidance documentation has been produced for use by system integrators and component suppliers across industry, and government staff responsible for the acquisition of PYRAMID-compliant products [PYRA].

Both modularity and the use of open architectures were defined as critical enablers to achieving the goals of PYRAMID by the UK MOD. Interestingly, a PYRAMID compliant Architecture Framework has been created that incorporate the FACE Technical Standard [PYRB].

European Component Oriented Architecture (ECOA®)

The European Component Oriented Architecture (ECOA) is an open specification for a software framework for mission system software comprising components that are both realtime and service-oriented. The objective of the research is to reduce the cost and timescales for production and modification of complex real-time aircraft software systems, by facilitating software portability and reuse. The technical solution for the application-level architecture framework was developed from a number of contemporary software development techniques that were deemed to be useful to the target problem domain, predominantly the Service Component Architecture (SCA). This provides flexibility whilst still retaining the predictability needed to achieve the rigorous safety certification and security accreditation processes used in the defense industry. The software architecture supports new business models for software procurement which may radically change the way that the defense industry does business [ECOA].

The ECOA approach is different than the FACE Approach. The FACE Technical Standard focuses on application portability at OS level as well as on standardizing interfaces between all application layers in an avionics system through a mediation layer. It also aims to provide a “reference domain” data model for a functional interface in military mission systems. Subsequently, ECOA and FACE are complementary technologies, and since FACE is an interface-oriented standard while ECOA is a component-oriented standard, it is possible to execute ECOA applications on a FACE implementation [ECOW].

Modular Software Design, Verification, and Certification Challenges

While the goals and benefits of modular software are desirable, the complexity of achieving the goals and benefits introduces several challenges. For example, the need for the software component to be functionally correct (without defects), safe, and secure becomes paramount. If a defect or vulnerability is discovered in a fielded software component that is used in a few systems, then those need to be brought out of service and remediated. This comes at a cost and may be an inconvenience. However, remediating a defect or vulnerability in a software component used in most commercial airplanes or in hundreds of thousands of vehicles comes at a much higher cost and may have consequences beyond inconveniences.

Another challenge is to design the software component in such a way that it can be widely reused – for example in a context where there are different levels of safety and security criticality. This is particularly challenging because safety standards like DO-178C specify the certification process for an entire software system and may require certification of the software running on the operational hardware. Modular software design emphasizes the independence of system components, each requiring their own certification process. This independence can present a challenge, as integrators may need to certify a complete system composed of modular software components, including those originating from different vendors.

In the case of a FACE Unit of Conformance (UoC), there is the challenge of developing the UoC so that it complies with FACE requirements for multiple Operating Systems Segment (OSS) profiles and capability sets. The FACE Reference Implementation Guide, Version 2 includes the following guidance: "FACE UoCs are particularly likely to see use in new contexts that were not foreseen by their developers. It is possible that a UoC provides functionality that has safety implications in a new integration that are higher than those provided in the initial system. To ensure the broadest audience for FACE UoCs, a developer should consider, and evaluate the expense of, qualifying their software at higher levels of rigor [G240]."

Best Practices for Developing Safe and Secure Modular Software

Fortunately, the standards presented earlier in this paper do provide good guidance to help overcome the challenges of developing safe and secure modular software. In the section of the paper, we'll explore some best practices that complement the guidance in the standards

System context

As previously mentioned, one of the challenges with the re-use of modular software components is ensuring they can be 'plugged into' a new environment. This usually requires both an understanding of interfaces and functionality – the interfaces provide the means to communicate with the component, and the functionality specifies how the component relates the outputs to the inputs. Traditionally, interface design documents and software requirements documents have been used to capture the interfaces and functionality of a software component.

While these documents are useful, modular software design also relies on an architecture and a platform. The FACE Technical Standard – for example – does specify a reference architecture that is useful for ensuring interface consistency. For the functionality, the platform provides a framework within which the software component will function. Model Based Systems Engineering (MBSE), and the use of the Object Management Group's Systems Modeling Language (OMG SysML®) provide a means for developing a platform model to support the development and reuse of modular software components.

MBSE has been used to develop the FAF (Future Vertical Lift (FVL) Architecture Framework) and has also been used to develop the GCIA (Ground Combat Systems (GCS) Common Infrastructure Architecture). The use of MBSE in the FAF and GCIA provides a means of specifying hardware and software components within the context of a broader enterprise architecture. Both FAF and GCIA models are aligned with the FACE Technical Standard.

Requirements traceability

Of specific concern with re-using modular software components is the ability to comply with functional safety and security standards. These standards require safety and security analysis to be done at the systems level, and for certification activities to be in the context of the operational system. As the level of risk or criticality of the software increases, so does the rigor of the testing and verification that must be done to assure the software is sufficiently safe and secure. ARINC 653 provides means to isolate software components, and in some cases that can allow software components to be verified independently.

However, standards like DO-178C still require full traceability of requirements to source code (and in some cases object code), requirements to test cases, and test cases to source code – and test results must be recorded to show that all tests have been passed successfully, and all the source code (and in some cases object code) has been executed by the test cases.

When developing a modular software component, it's important for the design artifacts to include the software requirements (both high-level and low-level), the test cases, the source code, and all the traceability information across the artifacts. When re-using a modular software component, the high-level software requirements must be traced to system (or sub-system) requirements. Establishing this traceability may uncover gaps where the context for the component may not fully use all of the functionality of the component. In that case, the integrator must carefully assess and document any safety or security risks associated with the code that no longer traces up to system requirements. Including the software requirements (both high-level and low-level), the test cases, the source code, and all the traceability information across the design artifacts as part of the design package for modular software component makes the job of resolving any requirements gaps straightforward. Examining the top sections of Figure 2, the system requirements are shown in the top left section, high-level requirements are shown in the next section to the right, low-level requirements to the right of the high-level requirements, and source code functions in the section all the way on the right hand side. Consider the case where an integrator is incorporating a modular software component into a new application, and they have linked the system requirements to the high-level requirements of the modular software component. All the high-level requirements can be selected and then following the traceability links created by the modular software component developer, it is easy to see if any low-level requirements or software functions are not linked. Any gaps will need to be examined and corrected.

As seen in the figure, the gap analysis shows there are 6 software functions that do not trace to low-level requirements. A similar approach can be applied to identify test cases that are not linked to requirements and/or software functions. In addition to selecting elements in the traceability tree, the ability to navigate upstream and downstream through traceability links enables the integrator to quickly explore and examine all the traceable artifacts of interest.

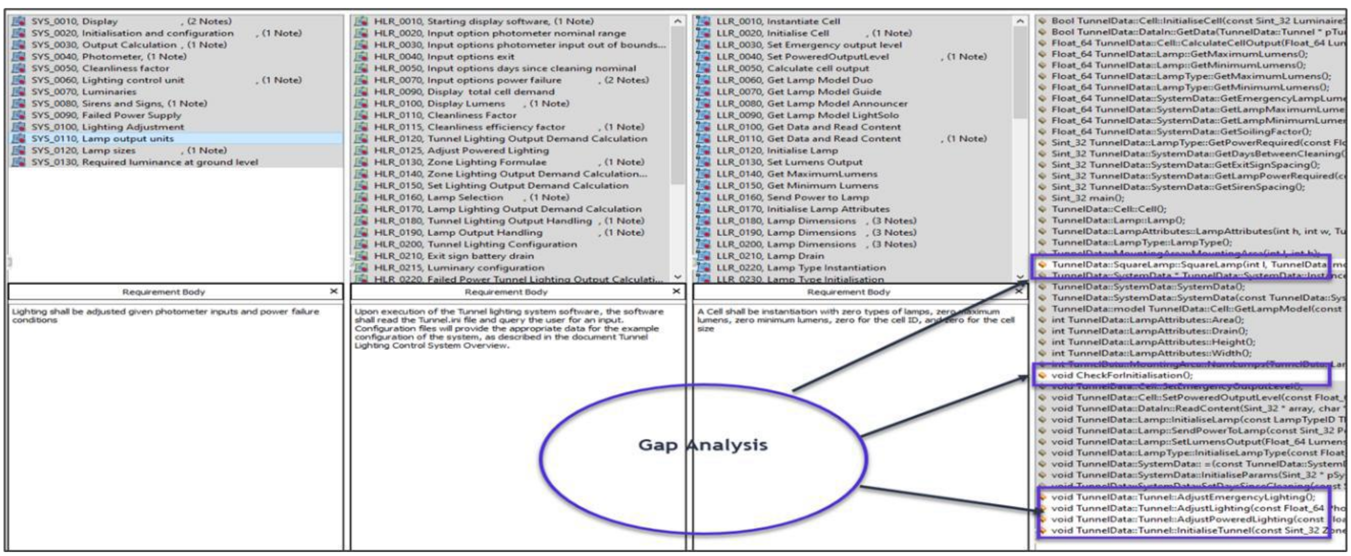

Figure 2 - Requirements, code, test case traceability

Impact analysis and change management

In some cases, integrating modular software components into a new context uncovers a need to make some changes to the requirements and tests. For example, there are some parameters for an algorithm that require slightly different values in the new context. Here too, including all the traceability information across the design artifacts as part of the design package for modular software component makes the integrators job easier.

Traceability information provides a visual impact analysis to see what source code and test cases may be impacted by a change in requirements. Where there may be more than one option for making a change to the source code in response to a change in requirements, the impact analysis provides valuable information to make more informed decisions on which option to choose.

Once a design change is to be made, the traceability information is used to identify up/down stream items affected by a change. Integrators use the traceability information to guide the changing of the source code and test cases (as appropriate), and then the test cases can be run to ensure the correctness of the change. Looking at 2 example low-level requirements shown in Figure 3, both the upstream and downstream requirements are listed (note that since these are low-level requirements, there are no downstream impacts). The status of both requirements has been highlighted to show it has changed, indicating that the test cases have been regressed, and different test results have been obtained. For LLR-0460, there is also an indication that the contents of the requirement have also changed.

LDRA TBmanager Project Baseline Analysis Report

Baseline	Current Project against **Pre-Shipment Baseline_012020092514**
Comparison Criteria	Requirements Only

Modified Requirements

Number	**Name**	**Body**
LLR_0100	Get Data and Read Content	Get Data shall parse the tunnel lighting system initialisation data using a provided initialisation file

Upstream Impact

Number	**Name**	**Body**
HLR_0215	Luminary configuration	Luminaries shall be grouped in sets of one, two, or three depending on where they lie in the tunnel. Lamps within a set can be used in any combination to assist in the optimisation process, which means that the control system need have no concept of the discrete luminaires themselves, only of the sets to which they belong. Luminary grouping will be described in the configuration file
HLR_0010	Starting display software	Upon execution of the Tunnel lighting system software, the software shall read the Tunnel.ini file and query the user for an input. Configuration files will provide the appropriate data for the example configuration of the system, as described in the document Tunnel Lighting Control System Overview.
HLR_0200	Tunnel Lighting Configuration	Configuration facilities shall be provided to facilitate the automatic calculation of the number of lamps which may be positioned in a particular tunnel ceiling area

Downstream Impact: None

Modification

Content Changed	**Source Mapping Changed**	**Upstream/Downstream Changed**	**Status Changed**
No	No	No	Yes

Number	**Name**	**Body**
LLR_0460	Adjust Powered Lighting	Given the photometer input in mAmps and powered conditions, the lighting output demands shall be adjusted across all zones. Cross check against the other powered conditions

Upstream Impact

Number	**Name**	**Body**
HLR_0125	Adjust Powered Lighting	Given the photometer input lighting shall be calculated across all zones
HLR_0237	Photometer signal conversion	Photometer measurements shall be converted into an output signal of 4-20m/A (directly proportional to the luminance measurement (i.e. 4.0 m/A being given as zero luminance and 20 m/A as 6,500 cd/m2) for signal transmission to the Lighting Control Unit

Downstream Impact: None

Modification

Content Changed	**Source Mapping Changed**	**Upstream/Downstream Changed**	**Status Changed**
Yes	No	No	Yes

Figure 3 - Impact analysis and change management

Test automation

Test automation provides a means to save time and improve quality both during the development of modular software components as well as when integrating modular software components. Modern software development utilizes continuous integration often provided through a DevSecOps platform to automate the integration and testing of software.

During the development of modular software components, static analysis can be used to ensure compliance with commonly used industry code standards such as MISRA, CERT, and CWE (Common Weakness Enumeration) [MISRA][CERT][CWE]. Note that while CWE is more than a code standard, many CWE weaknesses include implementation guidance that can be checked using one or more coding rules.

Static analysis can also be used to check compliance with some of the FACE Technical Standard requirements that constitute a code standard. For a C++ UoC, one of the primary requirements of interest in the FACE Technical Standard is found in section 4.2.3.3. This is a complex requirement that includes many restrictions on the use of C++ language features – including variations by profile (capability set). Static code analysis can examine the code for the use (or absence) of programming language features, including multi-threading constructs. Figure 4 shows a summary report of the violations for a sample source code file, indicating which procedures are in compliance with the FACE Technical Standard requirements that constitute a code standard.

Code Review Result	Procedure	Source File	Unique Violations	Failure Density (Viols/R.Line %)
PASS	BALSA::Aircraft_Config::TypedTS::TypedTS	BALSA_Aircraft_Config_TypedTS.cpp	0%	0%
PASS	BALSA::Aircraft_Config::TypedTS::Set_Reference	BALSA_Aircraft_Config_TypedTS.cpp	0%	0%
FAIL	BALSA::Aircraft_Config::TypedTS::Receive_Message	BALSA_Aircraft_Config_TypedTS.cpp	11%	20%
FAIL	BALSA::Aircraft_Config::TypedTS::Send_Message	BALSA_Aircraft_Config_TypedTS.cpp	11%	20%
PASS	BALSA::Aircraft_Config::TypedTS::Register_Callback	BALSA_Aircraft_Config_TypedTS.cpp	0%	0%
PASS	BALSA::Aircraft_Config::TypedTS::~TypedTS	BALSA_Aircraft_Config_TypedTS.cpp	0%	0%

Figure 4 - Static analysis results (summary)

Complementary to static analysis, the FACE Conformance Test Suite (CTS) automates the process of ensuring FACE UoCs are using the correct interfaces based on their data model and designated profile. While the FACE CTS does not perform any functional testing of a FACE UoC, software suppliers seeking FACE Conformance must provide verification evidence to prove appropriate functional testing of the UoC has been successfully completed.

In addition to static analysis, DevSecOps platforms can be used to perform additional testing, including requirements-based testing, measuring structural code coverage, and robustness testing. Consider again the case of an integrator is incorporating a modular software component into a new application. After aligning and amending the high-level and low-level requirements as needed, the software component must undergo unit testing followed by integration testing and verification. Depending on applicable level of safety and security, an appropriately rigorous course of requirements-based testing, robustness testing, and structural code coverage analysis must be successfully completed. For the highest levels of safety and security, additional testing may be required.

Creating and managing certification evidence

An important consideration for any safety and security critical software system is attaining certification to appropriate safety and security standards. This is even more important when considering modular software components where certification provides confidence that the components are portable and reusable. A case in point is that one of the principles of MOSA is to certify conformance.

Certification and conformance rely heavily on following a set of defined processes, completing appropriate activities, and recording and manging evidence that the processes have been followed and the activities successfully completed. In the case of modular software components, the component developer can provide some of the of the certification evidence, but some will have to be created by the software integrator. The software developer can provide artifacts for the software integrator to use in creating certification evidence. Figure 5 shows an example of a set of reports that can be used as evidence to achieve DO-178C certification.

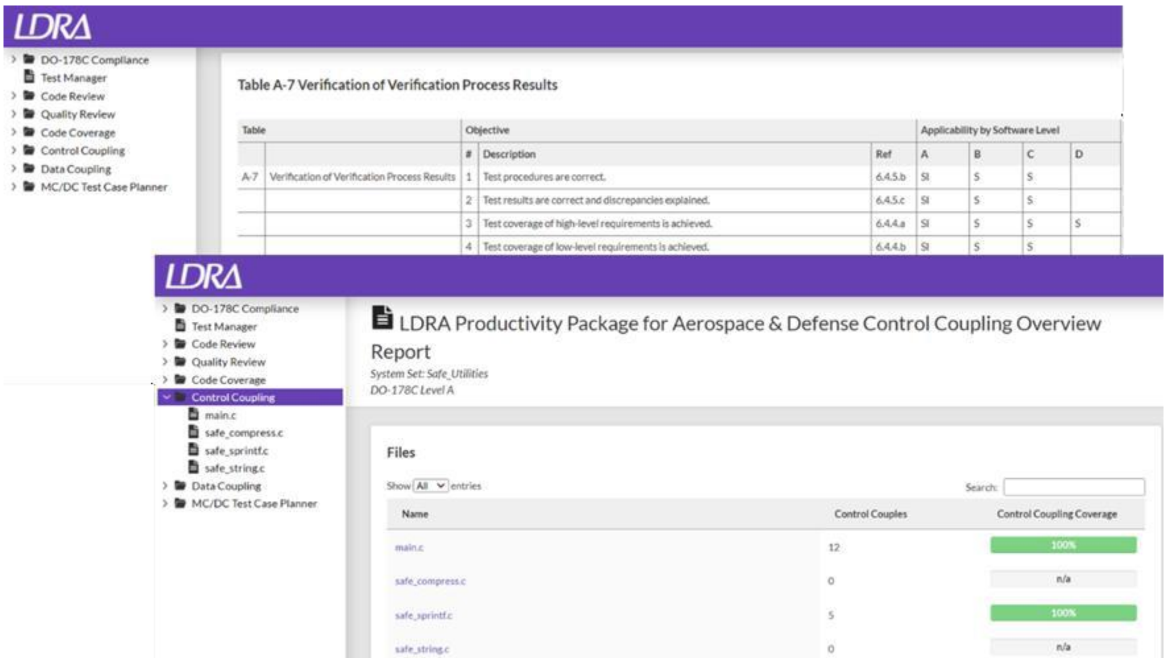

Figure 5 - Certification evidence

Summary

The use of modular software presents both opportunities and challenges, particularly in the context of safety assessments as outlined in standards like DO-178C. This paper has explored standards available today that support modular software, challenges with the development and integration of modular software, and has provided practical guidance on overcoming these challenges.

A significant challenge with the use of modular software is to ensure interoperability and safety across independently developed components. This challenge is combined by the need for adherence to evolving standards and the complexities of lifecycle management. To address these issues, this paper has highlighted best practices, including adherence to open standards like the FACE Technical Standard, the importance of requirements traceability, and the implementation of continuous integration and automated testing practices.

Referenced Documents

(Please note that the links below are good at the time of writing but cannot be guaranteed for the future.)

C232 FACE® Technical Standard, Edition 3.2 (C232), published by The Open Group, August 2023; refer to: www.opengroup.org/library/c232

WM1 The Evolution of Modular Construction: A Historical Perspective refer to: https://www.whitleyman.com/post/the-evolution-of-modular-construction-a-historical-perspective

IS1 Icon of the Seas (38545). Vessel Register for DNV. Refer to: https://www.royalcaribbeanpresscenter.com/fact-sheet/35/icon-of-the-seas/

MD1 Tseng, M.M., Wang, Y., Jiao, R.J. (2018). Modular Design. In: The International Academy for Production (eds) CIRP Encyclopedia of Production Engineering. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-35950-7_6460-4

DO297 RTCA DO-297. Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations, RTCA, Inc., 2005.

AC170 Federal Aviation Administration (FAA), Advisory Circular AC 20-170 Integrated Modular Avionics Development. Verification, Integration and Approval using RTCA/DO-297 and Technical Standard Order C153, U.S. Department of Transportation, Oct. 28, 2010.

DO178 RTCA DO-178C. Software Considerations in Airborne Systems and Equipment Certification, RTCA, Inc., 2011.

AC148 Federal Aviation Administration (FAA), Advisory Circular AC 20-148: Reusable Software Components, U.S. Department of Transportation, Dec 7, 2004.

A653 "Avionics Application Software Standard Interface: ARINC Specification 653 Part 0". Aeronautical Radio, Inc. Dec 23, 2019.

POSX "IEEE 1003.1-2024". IEEE/Open Group Standard for Information Technology--Portable Operating System Interface (POSIX™) Base Specifications, Issue 8. IEEE Standards Association. Sep 5, 2019.

MOSA Modular Open Systems Approach. Office of the Under Secretary of Defense, Research and Engineering. Refer to: https://www.cto.mil/sea/mosa/

PYRA PYRAMID Technical Standard and Guidance, UK Ministry of Defence. Refer to: https://www.gov.uk/government/publications/pyramid

ECOA European Component Oriented Architecture (ECOA®) Collaboration Programme. Refer to: https://www.ecoa.technology/index.html

PYRB PYRAMID for avionics and mission systems Briefing and Q&A Session 14th November 2024, https://assets.publishing.service.gov.uk/media/6735bf1e54652d03d51610ea/20241114-PYRAMID_Launch_Webinar-O.pdf

ECOW	European Component Oriented Architecture (ECOA®) Collaboration Programme: ECOA White Paper, Prepared by Dassault Aviation further to 2017-11-15 version prepared with BAE Systems (Operations) Ltd
G240	Reference Implementation Guide, Version 2, for FACE® Technical Standard, Edition 3.0 (G240), published by The Open Group, January 2024; refer to: http://www.opengroup.org/library/g240
MISRA	MISRA Coding Standards. MISRA. Refer to: https://www.misra.org.uk/
CERT	SEI CERT Coding Standards. Carnegie Mellon University Software Engineering Institute. Refer to: https://wiki.sei.cmu.edu/confluence/display/seccode/SEI+CERT+Coding+Standards
CWE	Common Weakness Enumeration; refer to: https://cwe.mitre.org/

About the Author(s)

Alex Lim – Alex serves as a Multicore Lead and Principal Field Application Engineer with LDRA. Over the years he has worked closely with industry leading companies in automotive, aerospace, and other safety critical domains. Alex has driven innovative solutions with LDRA's customers in ADAS systems, helped RTOS and silicon vendors achieve safety and security goals, and worked closely with avionics suppliers to meet the latest standards. Alex also works with LDRA customers and distributors to bring LDRA solutions and international safety and security standards to emerging markets. He represents LDRA on industry bodies and has presented at numerous events including autonomous vehicles conferences and the Digital Avionics Systems Conference (DASC). Over the course of his career, Alex worked at the Space and Missiles System Centre on safety and mission critical Flight Safety and Flight Management software, and designed autopilot simulations for UAVs.

Stephen Di Camillo – Steve is the Technical Marketing and Business Development Manager with LDRA USA. Steve has over 30 years of experience in embedded systems and software development, focusing on model-based engineering and standards compliance. He has worked with many customers around the world in the automotive, aerospace, rail, medical, telecommunications, energy, and industrial equipment industries. Throughout his career, Steve has helped engineering organizations to adopt new processes, tools, and technologies to improve the efficiency and effectiveness of their systems and software development and testing efforts while maintaining compliance with evolving and emerging technical standards.

About The Open Group FACE® Consortium

The Open Group Future Airborne Capability Environment® Consortium (the FACE® Consortium), was formed as a government and industry partnership to define an open avionics environment for all military airborne platform types. Today, it is an aviation-focused professional group made up of industry suppliers, customers, academia, and users. The FACE Consortium provides a vendor-neutral forum for industry and government to work together to develop and consolidate the open standards, best practices, guidance documents, and business strategy necessary for acquisition of affordable software systems that promote innovation and rapid integration of portable capabilities across global defense programs.

Further information on the FACE Consortium is available at www.opengroup.org/face.

About The Open Group SOSA® Consortium

The Open Group SOSA® Consortium enables government and industry to collaboratively develop open standards and best practices to enable, enhance, and accelerate the deployment of affordable, capable, interoperable sensor systems. The SOSA Consortium is creating open system reference architectures applicable to military and commercial sensor systems and a business model that balances stakeholder interests. The architectures employ modular design and use widely supported, consensus-based, nonproprietary standards for key interfaces.

Further information on the SOSA Consortium is available at www.opengroup.org/sosa.

About The Open Group

The Open Group is a global consortium that enables the achievement of business objectives through technology standards and open source initiatives by fostering a culture of collaboration, inclusivity, and mutual respect among our diverse group of 900+ memberships. Our membership includes customers, systems and solutions suppliers, tool vendors, integrators, academics, and consultants across multiple industries.

The mission of The Open Group is to drive the creation of Boundaryless Information Flow™ achieved by:

- Working with customers to capture, understand, and address current and emerging requirements, establish policies, and share best practices
- Working with suppliers, consortia, and standards bodies to develop consensus and facilitate interoperability, to evolve and integrate specifications and open source technologies
- Offering a comprehensive set of services to enhance the operational efficiency of consortia
- Developing and operating the industry's premier certification service and encouraging procurement of certified products

Further information on The Open Group is available at www.opengroup.org.