THE Open GROUP

Developing Portable/Reusable Certification Artifacts

An approach to reducing the effort needed to add portable capabilites to a system supporting FACE conformant software

US Army Aviation FACE™ TIM Paper by:

Author Name Christopher J. Edwards, PMP,
UH-60 CMS Systems Engineering Lead,
AMRDEC FACE Verification Authority SME

January 2016

Distribution Statement A - Approved for Public Release
- Distribution Unlimited (control number PR1919)

Table of Contents

Executive Summary

Traditionally, avionics software requirements stem from aircraft platform needs. These requirements flow through a system specification, then into the software requirements, designs, test procedures, and source code. The artifacts are written with the system in mind. Often the name of the system is repeated everywhere, and the trace for coverage is developed and maintained stemming from the system-level requirements. This pattern must be broken to produce reusable artifacts.

For Open System Architecture (OSA) strategies to capitalize on portability and reuse opportunities, the manner in which we trace requirements and document the needs for our portable software must change.

The Crew Mission Station (CMS) Project was started to create a system for hosting software ported from the original development, and intends on hosting Future Airborne Capability Environment (FACE™) Conformant software developed without the CMS system specifically in mind. In the CMS project, a new approach to documentation was developed to avoid the increased work in porting these capabilities.

Developing a System and Modular Capabilities

Shifts towards the development and marketing of generic-reusable system components is reducing cost across the hardware and software industries. Instead of a single system designed for a single use, systems are developed for general use and software is being developed independent of the specific system.

In the 1990s it wasn't uncommon for engineers to have both a cell phone and a calculator in their pockets. Each of these devices was custom built for a single function. Advances in technology have enabled us to combine several functions into one device by creating a generic computing system. The software for these computing platforms is developed to the requirements of the software and later integrated with the hardware. The mobile devices of today are not built with knowledge of the software enabled capabilities they support. The software is designed to the requirements of the capability, with the potential hardware systems factored in.

The Future Aviation Capability Environment (FACE) approach and other Open Systems Architecture (OSA) approaches are being developed and utilized to bring these advantages into the aviation industry.

Within the aviation industry, we still have development patterns held over from the single system requirements pattern. Our documentation stems from the source requirements of the complete, overall system. The intended use and program name are pervasive in our work products and artifacts we produce for flight qualification. Figure 1 shows a representative document tree.

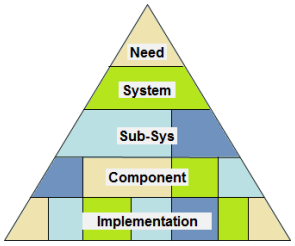

Figure 1- Document Tree

For flight qualification at the middle and higher assurance levels, every line of software code must have a source requirement, and all of those source requirements must trace back to an aircraft level need. Figure 2 shows the trace lines to the source requirements over the representative document tree.

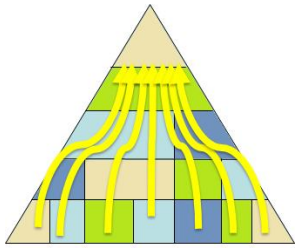

Figure 2- Trace to Source

The avionics industry has developed patterns to meet these high standards. We capture high level requirements for a system, and when we add capabilities, we iteratively add to those high level documents. When developing the capability, all of our artifacts tend to trace directly back to the high level requirements, and the resulting language is pervasive in the artifacts we provide for flight qualification. Figure 3 shows a representation of adding capabilities to an already developed system. These patterns were clearly not developed with the thoughts of independent capabilities in mind.

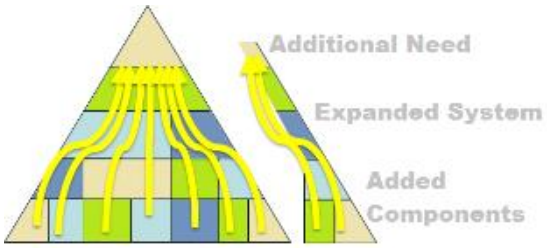

Figure 3- Adding Functionality

Developing Portable/Reusable Certification Artifacts

One of the key FACE quality attributes is portability. To achieve portability, we must develop our software for reuse across platforms. We must also develop platforms that account for future need. In the cell-phone industry the computing system still derives some of its requirements from intended use (such as a camera, or a stereo headphone output). The software derives some of its requirements from the intended system (operating system, availability of GPS data). In the avionics industry, we cannot fully adopt the patterns used by the cell phone industry and mobile application developers. The required artifacts for certification are not there; the need to derive from a system requirement does not exist for mobile applications.

Following the traditional patterns of documenting such a system for flight qualification will lead to increased costs unless new patterns for documentation are established. This leaves the question of how we can capture a software requirement that did not stem from the specific aircraft need. We need to provide the documentation stemming from the need for a capability beyond the aircraft need. We also need to provide a means to capture a system that supports future capabilities.

FACE Conformance Artifacts

Conformance to the FACE Technical Standard is captured at the Unit of Conformance (UoC) level. In the notional diagrams used above, the UoC is represented by the bottom two layers (Component and Implementation). The Conformance program often uses artifacts from higher layers as evidence of conformant behavior.

Once the UoC passes conformance it is referenced in the FACE Registry. From there the UoC can be identified and reused on other programs.

When the UoC is used on another program, what is done with these higher level artifacts? The new program can't use them as they are.

The information contained in these documents must be ported to new documents in the new target system.

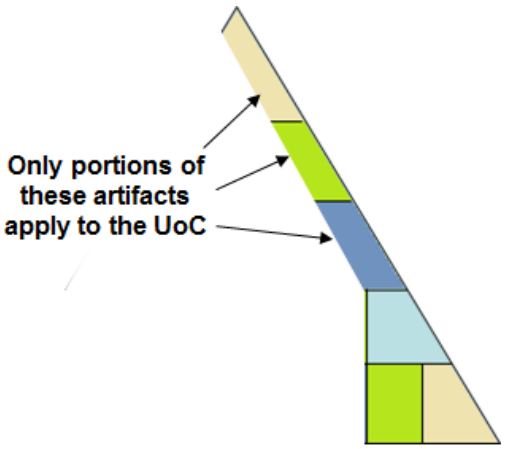

Figure 4- FACE Conformance and Artifacts

Porting Qualification Artifacts

When a software package is developed as part of system development, the origin of the requirements, the requirements themselves, as well as the designs are all tied to the system specification. Pulling the documentation out of the origin system to place it in another system can lead to extra work.

As the level of criticality of the system function increases, so does the detail of the required artifacts. One of the core principles in these artifacts is proof of coverage – we must provide links (or traceability) to show that the requirements are derived from source requirements, have been tested and (at higher levels of criticality) that every part of the source code is required to meet the system intent.

When flight qualification does occur, it is completed for the aircraft hosting the system. All artifacts for all the software in the avionics systems are provided, and linked back to the aircraft requirements. When a capability is transferred from one system to the next these links will have to be reformed to the new aircraft.

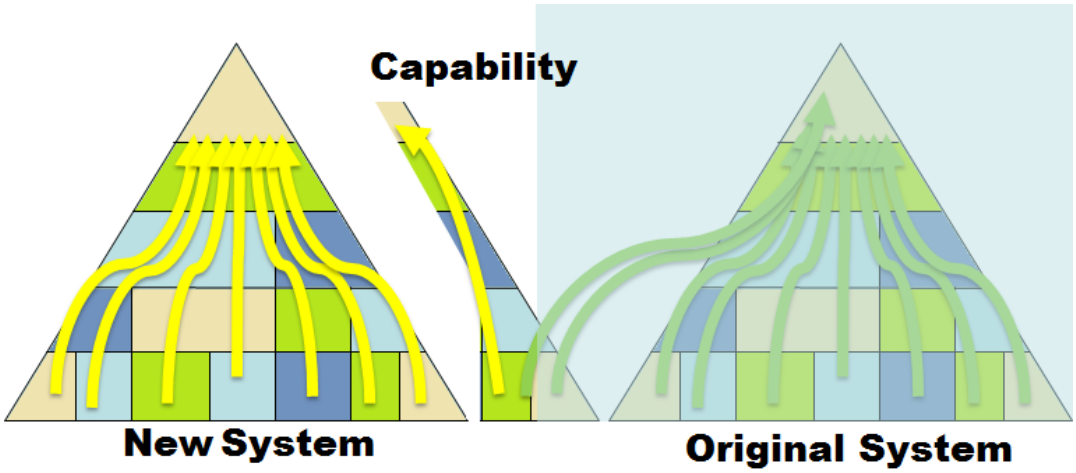

Figure 5- Documentation Porting Issues

This port of trace leads to several difficulties. There could be aspects of the capability that can't be fully traced to the new platform needs. The artifacts being transferred were probably written with the original system in mind, leaving the name of the system repeated everywhere. This is the pattern that we must break to produce reusable artifacts.

UH-60 Crew Mission Station: A System and Its Portable Capabilities

A System Supporting FACE Conformant Software

The UH-60 Futures Team started the Crew Mission Station (CMS) project with the objective of providing increased situational awareness (SA) for the non-rated crew members in the cabin of the aircraft. The resulting project vision, "To produce an enduring open systems architecture and management plan capable of providing new capabilities to the H-60 fleet in the shortest timeframe possible" provides the overarching direction for project design and development.

CMS is providing visual displays hosting SA and other mission related capabilities including but not limited to:

- Digital moving map
- Mission flight plan and progress
- Fuel management calculations
- External video image such as onboard cameras or unmanned vehicles
- Aircraft systems status and crew alerting (EICAS)
- E-Reader for viewing checklists, approach procedures, still imagery, and other digital files

The CMS design, accompanying open architecture and management plan will provide an enduring method to provide additional, as well as currently undefined capabilities, to the H-60 crew more rapidly than is currently possible.

Being capable to support FACE conformant software is a requirement on the CMS project. When examining this requirement combined with the vision of rapid deployment, it became clear that the CMS project must look at the considerations and impacts of adding new capabilities in the shortest time. For example:

- What sort of documentation modifications would have to be made?
- How would the system designs be affected?
- How would the system integration testing be affected?
- How are the flight qualification artifacts managed?

Considerations for the CMS project:

- Any new capability added to CMS from another platform should come with its own artifacts.
- Do not include any artifacts that specified the name of the other platform.
- Avoid the work of separating the capability CMS adds from artifacts that cover more than just that capability.
- Do not add requirements into a top level document that would drive an extensive need for re-testing.

Developing Portable/Reusable Certification Artifacts

As the CMS team considered how it would like to receive documentation for inclusion into CMS, it was realized that CMS should provide the courtesy of developing our requirements in a way that would benefit the receiver of our artifacts.

Precedence for a Solution

Within the avionics industry we have already developed patterns for incorporating artifacts from a prior development.

Hardware Reuse

The requirements for a truly off-the-shelf display would not reference the project it is installed on. The project would establish the need for a display and include requirements the display must meet. A display would be selected and an analysis would be performed to show the display met the requirements.

The system integration would prove that the display is capable of supporting the new software. The display comes with its own development and test artifacts that do not need to be recreated for the new project.

Software Purchase

When a software package is purchased to provide a solution, the requirements and design are separated out of necessity. The name of the destination system was generally not known at the time of development. The trace into the subsequent system is performed through requirements that describe the need for the reusable component and documentation is produced for the integration of the previously developed component into the new host system.

This represents one way to add software capabilities that do not inherently include full trace to the parent program. The requirements trace in these software packages include software implemented in a portable manner, reflective of some design decisions and implementation details that are not strictly traceable to a single aircraft.

Integrated Modular Avionics

DO-297 “Integrated Modular Avionics (IMA) Development Guidance and Certification”, along with AC 20-170 provide a qualification path through the FAA for developing a system using software components with independent qualification efforts.

Through the use of IMA, software capabilities can be developed in a portable way and have a certain level of independent qualification. IMA also allows the host system to achieve a level of qualification prior to the addition of new capabilities.

These precedents can be used to create portable artifacts within the same project as the initial host system. The initial system should treat each portable capability as if it was separately procured. Each separate capability should derive from the specific requirements for that capability, not from the specifications for an entire aircraft or aircraft system.

Documenting a Portable Capability

If the CMS project is to allow for the incorporation of FACE conformant software developed for other projects, it would be best if the incorporated artifacts could be seamlessly integrated into the CMS document tree without affecting the artifacts themselves.

For this to work, the original development would have to be documented without making extensive references to its origin system. The software designs would have to be developed in their own independent models. The requirements would have to independently derive from the origin requirements and the test procedures would have to be developed without relying on other parts of the system.

Figure 6 shows a notional representation of the documentation needed for an independent capability. The capabilities should be developed following the IMA guidance, including all artifacts listed in DO-297 for Application Acceptance (DO-297 Section 4.3, including related artifacts in section 4.2).

Included in this diagram is the notion that the Portable Capability should have Functional Test Cases developed that can be fed into a System Integration Test Procedure when the Capability is integrated into a target system.

If the process of incorporating FACE UoCs into the CMS System proves to be easier if these patterns are followed, the CMS system should document its own capabilities in a similar manner.

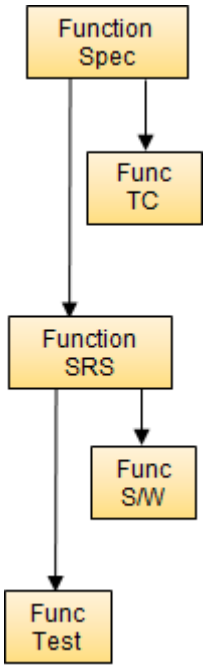

Figure 6- Portable Capability

Defining the System

As capabilities are added to the CMS System, minimizing the extent of the documentation affected should be a goal. The CMS project should limit the impact of a change in order to minimize the amount of retesting generated by the addition. By generating documentation for the physical system and its software infrastructure independent of the capabilities it supports, most of the artifacts and testing associated with the infrastructure hardware and software would not need to change when new capabilities are added.

The CMS System will be documented as a computing system addressing the needs of potential software capabilities not yet defined. This system will be documented as a collection of hardware devices and software components, which may already be documented as independent components.

The CMS System documentation will include the requirement to provide a display and the basic requirements driving the selection of the display. The Transport Layer chosen will meet the requirements in the CMS System documentation placed on the Transport Services Segment (TSS). Each separately purchased component will have one or more requirements specifying the need in order to support procurement of the system components.

The proof that the selected components meet the system requirements will be developed for the CMS System independently of the artifacts for the Display, TSS, and other components purchased. The level of specific CMS System documentation needed is minor in comparison to the previously generated artifacts utilized from the development of the off-the-shelf components.

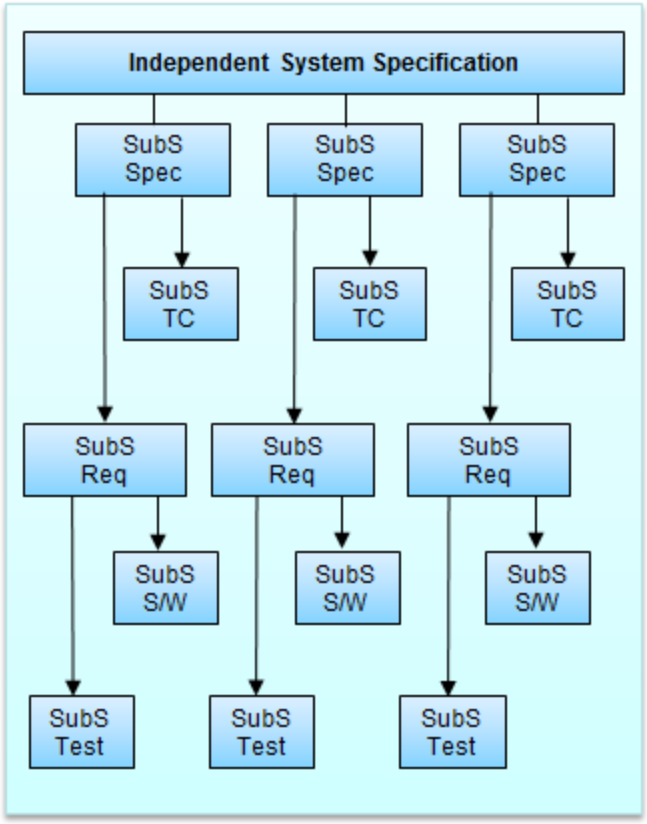

Figure 7- Independent System

Figure 7 shows a notional representation of the Independent System Tree. The documentation for this independent system should be developed following the IMA guidance, including all artifacts listed in DO-297 for IMA System Acceptance (DO-297 Section 4.4)

In addition to the IMA List, and included in Figure 7, is Subsystem Functional Test Cases developed for use in a System Integration Test Procedure when Capabilities are integrated into a fully realized system.

The CMS Vision Statement is "To produce an enduring open systems architecture and management plan capable of providing new capabilities to the H-60 fleet in the shortest timeframe possible." The vision for the CMS program is effectively to develop this independent system tree. Any capabilities added to this independent tree are used as proof of the concept.

The Need for an Integration Specification

The CMS System will still need to document the integration of these capabilities to cover requirements that stemming from the combination of the capabilities. The user interface for accessing each capability has to be documented. The partitioning of the system resources depends on the combination of all of the capabilities. There will likely be software that must be developed when the independent capabilities are eventually brought together into one system.

The UH-60 System Integration Specification is a high-level specification that often contains only a single requirement to reference all of the artifacts developed for one of the independent components of the system. This can take advantage of the same pattern used for incorporating off-the-shelf components in the CMS System design. This specification includes requirements like:

- *The UH-60 CMS System shall be a CMS System as identified by the CMS System Specification.*
- *The UH-60 CMS System shall include an EICAS Page per the AMRDEC H-60 EICAS Display Capability Specification.*

The integration specification also contains the specific requirements related to user access for each installed capability. The need for most Platform-Specific Device Services would be derived from this specification. It is the source document for the integrated system and treats each capability as independently developed.

The Integration Specification also supports the framework for an Integration System Test Procedure that uses the Independent Test Cases developed for each of the components included in the integration.

Figure 8 shows a notational representation of the Integration Tree and its relationship to the Independent Functions and the Independent System.

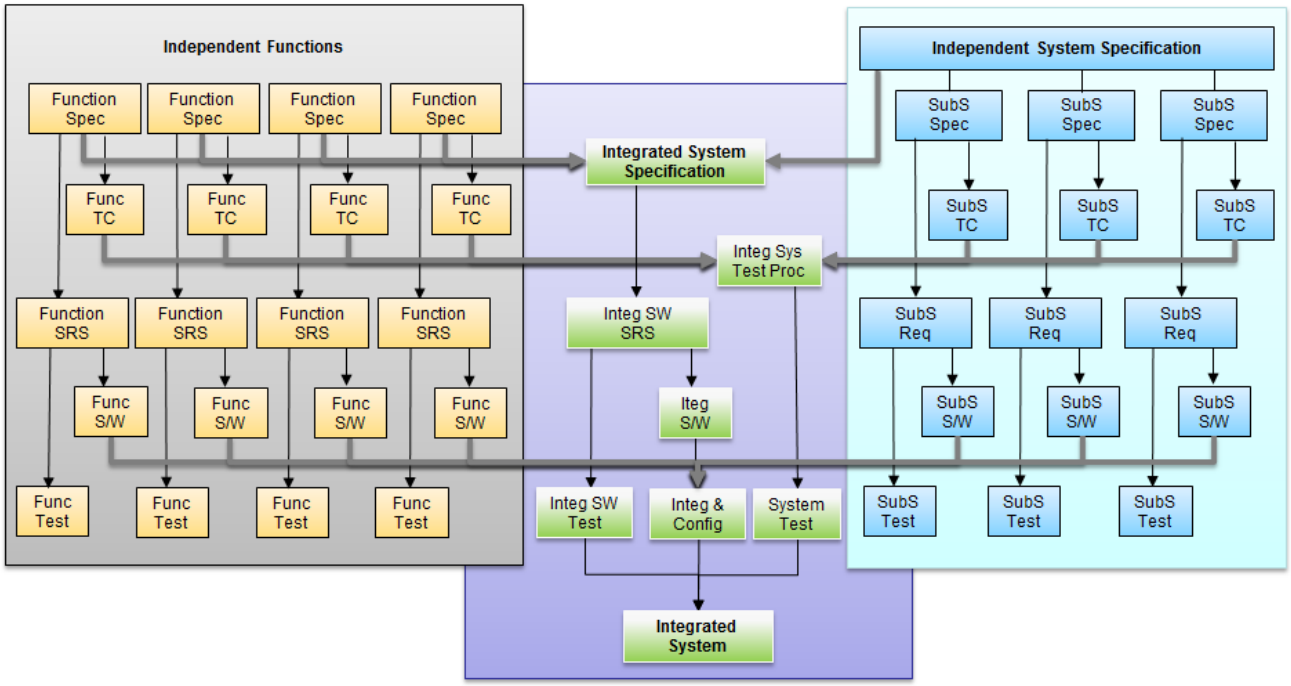

Figure 8- Integration Documentation

CMS and OSA

Part of the CMS project vision includes producing an enduring open systems architecture and management plan. The project is leveraging guidance from the Department of Defense definition of OSA as defined in the DOD, "Open Systems Architecture Contract Guidebook for Program Managers, Version 1.1, May 2013" and applying a Modular Open Systems Approach (MOSA) to achieve a "comprehensive" OSA consisting of:

- Government performed Functional Decomposition
- Government grouping of functionality (re-composition) to higher level modules representing competable, reusable capabilities. The resulting Reference Functional Architecture (RFA) will be based on Joint Common Architecture (JCA) functional decomposition.
- Government selection of a software and/or hardware technical reference frameworks (TRFs). Functions realized from the RFA are developed within the constraints established by a TRF. For CMS, the principal TRF is defined by the FACE Technical Standard.
- Government Allocation of functionality from the System Reference Architecture to hardware & software based on performance requirements, then defines the Objective Architecture of the system. The Objective Architecture can be defined and maintained by the government (which is CMS's choice) or the government can present Industry with the Full Reference Architecture plus the performance requirements of the system and allow a vendor to define an Objective Architecture and purchase the appropriate rights to configuration manage this solution..
- Separation of hardware and software modules enabling the Government to specify Government selection of Commercial-Off-The-Shelf (COTS) for certain hardware modules and allocate selected software capabilities to third party vendors
- Government Identification of "Key" interfaces between the modules defined by widely used industry standards (Ethernet IPv4, UDP, FACE I/O Services API)

The implementation of this comprehensive OSA approach will permit the accommodation of future capability upgrades and sustainment insertion in an expedient and efficient manner to meet CMS's technical and business goals. The OSA design will be technology agnostic, which will provide the flexibility and scalability to allow for new capabilities and future as-yet-unknown technology insertion.

The government will develop, own and manage the CMS requirements, functional/non-functional decomposition and the OSA and therefore not locked into a single vendor to develop and integrate new capabilities.

Independent Capability Software and a Reference Functional Architecture

The CMS approach for OSA starts with central management of shared functionality by defining and managing the capabilities and functional decomposition of a system. The deployment of these functional capabilities onto a system is then enabled by the design and development of standardized hardware and software technical reference frameworks and supported by a well-defined data rights strategy promoting competition and innovation throughout the system life cycle.

In order for CMS to achieve a workable OSA, step one is to perform a functional decomposition of the new capabilities into discrete core functions, which includes a description of the function, the required functionality, its behavior, and the data interoperability requirements. Since the resulting breakdown may result in capabilities defined at too low of a level, these core functions must then be recomposed or grouped, using business and technical decisions, to create more marketable/competable/reusable capabilities. The resulting reorganization is called a Reference Functional Architecture (RFA).

The functional decomposition being conducted by JCA will serve as the RFA for CMS and the specification level requirements for all the independent capabilities can be stored and managed in this RFA.

Selection of any Technical Reference Frameworks (TRFs) based on open architecture and life cycle goals, with the RFA comprise the System (Overall) Reference Architecture. Further architectural decisions to assign specific functionality to either hardware or software results in an Objective Architecture. This Objective Architecture design will be owned and maintained by CMS to ensure the end product is a standardized, modular OSA.

The FACE Data Architecture provides a high level method of describing data using a common language. The FACE Shared Data Model does not include a platform level model. A platform level model shared between components of a family of systems can be used to further detail interface requirements between components. This domain specific model can further define the requirements for a component without referencing a specific aircraft.

Developing Portable/Reusable Certification Artifacts

When an independent capability is developed from the RFA into components within the constraints of the Technical Reference Framework its resulting artifacts can be developed without referencing the specific aircraft or CMS system. When the Integration specification points to the RFA documentation as the source of its requirements, then the entire capability can have full trace to source requirements, regardless of the specific requirements for the intended aircraft. Using this pattern of development portable components can be developed with minimal impact when integrated to systems hosting those components.

Figure 9 shows a notional representation of the entire documentation structure. Each end system using the RFA picks the functions it needs from the RFA. The functions not selected by a target system do not become part of the artifacts used for qualification of that system and are not included in this notional diagram.

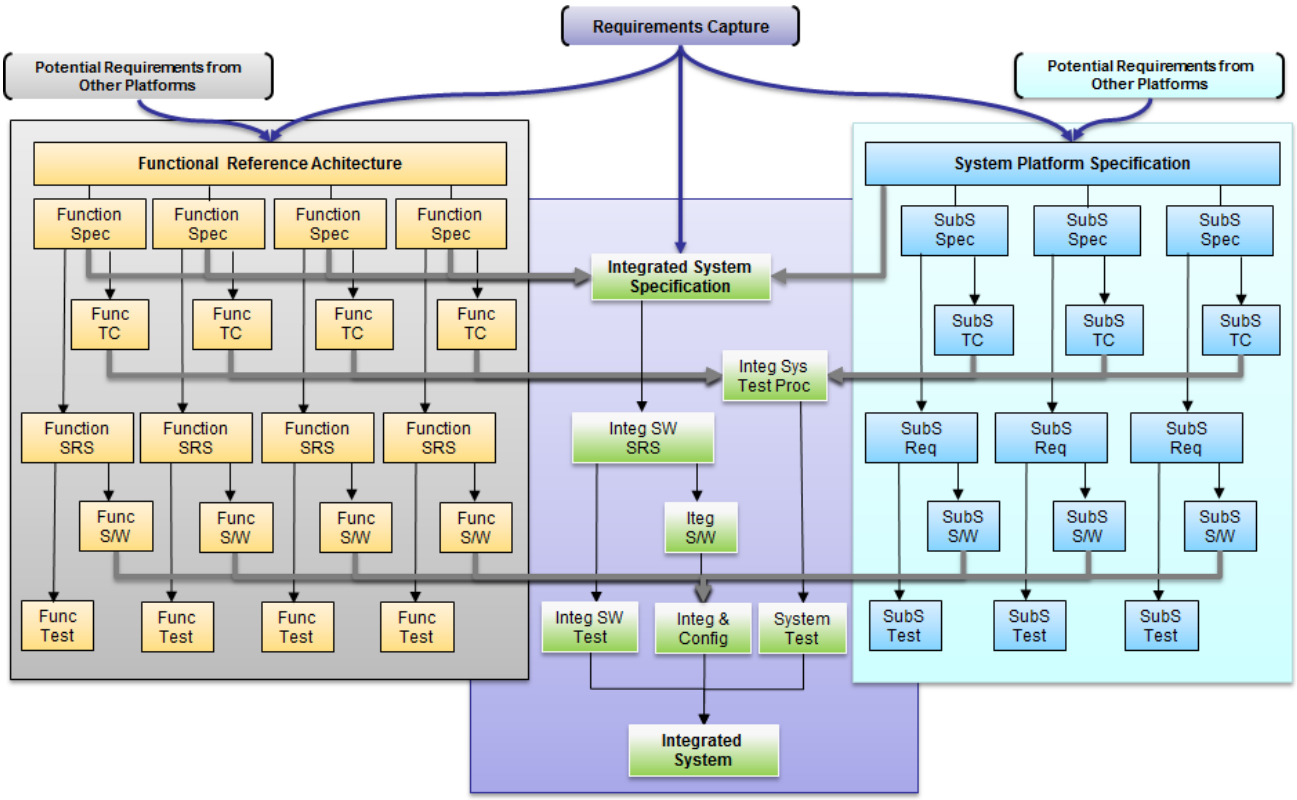

Figure 9- Full Artifact Flow

Overview of the Proposed Development Process

Stakeholder Requirements Capture

In the stakeholder requirements capture phase, the stakeholders are gathered to discuss the overall goals and high level requirements for the system. These requirements are gathered as outputs from the discussions and not tied to the specific system or subsystem.

Collect User Input

After the stakeholder requirements are captured, workshops with the potential users are conducted. The users are shown mock-ups of user interfaces generated from the stakeholder requirements meetings to gather feedback on what the system users want. These workshops also gather HMI data, such as workload impacts of different configurations of the software and hardware. The data is gathered in a report for the workshop.

Functional Decomposition

The information gathered from these two phases will be compared against the RFA to identify any functions already accounted for in the RFA.

If a function is accounted for the RFA, the stakeholder requirements and user input are compared to the data stored in the RFA as well as any portable software developed to that function. This could lead to the reuse, procurement, modification, or development of a capability under the RFA function. When some aspects of the collected requirements and user data will lead to a modification, the information captured will be provided as inputs to the modification by capturing additional requirements and design notes in the RFA and the modified software.

The portion of the captured requirements and user input that does not align with the RFA will be examined and could lead to new functions to add to the RFA. When new functions are to be added to the RFA the data captured is provided to the maintainers of the RFA for inclusion.

Other data captured might be included in the requirements for the base system, or in the requirements for the integration of the base system with the independent capabilities.

During this phase, all discovered activities will be traced back to their dependent system-level requirements. This is where the activity discovery process is especially leveraged to find incorrectly written or missing requirements, as well as derived requirements that will depend on them.

This allocation forms the basis for developing the independent capabilities.

Developing Portable/Reusable Certification Artifacts

Develop the Integration Specification

The Integration Specification captures the overall user interface and refers to the highest level documents for the system and each of its portable capabilities. The document will generally contain only one requirement for each independent capability (shall include a capability meeting the requirements for *X* as specified in *D*) and one requirement for meeting the independent system (shall include the software and hardware specified by the independent system specification *Y*).

Requirements for support components, such as Platform-Specific (PSS) Device Service, will be derived from this specification. The specific configuration of the configurable components is also derived from this specification. This document also serves as the basis for the System Integration Test Procedures that will include test procedures developed from Test Cases captured in each independent capability as well as in the Independent System.

Develop an Independent System

The independent system serves as the computational and user interface for the overall system. It includes the physical processers, displays, and networking support; as well as the software infrastructure components such as the Operating System, Input/Output Services (IOS), TSS, and Graphic Services. It also contains the basis for the overall user interface for the display, such as a menu system for accessing the installed capabilities.

CMS plans to use the stakeholder requirements, the user inputs from working groups, and the existing aircraft equipment as a guide to develop this documentation.

Historically, requirements documents are written to support 50% growth in processor, memory and additional I/O capability. An independent system may also have additional I/O or other hardware designed purely for supporting other new capabilities. When developing the independent system, CMS also reviewed other potential existing and future connectivity that may be required by future capabilities.

The development of the independent system should include high level models of the system functions. These models should be independent of any specific capability, but might include models related to generic use cases.

Development of the Independent System should include system test cases tied to the specification level requirements. These Test Cases will be developed into procedures as part of the integration.

Develop Independent Capabilities

Each Independent Capability identified should have its own high level requirements document derived from the stakeholder input and user working groups. When possible, this information should be captured in the RFA. All requirements for an Independent Capability should derive from the RFA, along with FACE and other frameworks and models that are not tied to the specific aircraft or system.

Part of the development of the Independent Capabilities will include models of the capability I/O and behavior. In keeping with the independence, these models should be developed independently of other models developed for the system.

Development of the Independent Capabilities should include system test cases tied to the specification level requirements. These test cases will be developed into procedures as part of the integration.

Developing Portable/Reusable Certification Artifacts

Derive Needed Integration Software

After developing the Independent System and the Independent Capabilities, the models for these components should be integrated under the Integration Specification. The integration will consist of making references to the other models and identifying missing components.

This integration should identify the need for PSS Device Service components for feeding data into the Independent Capabilities. The use of a specific hardware component as a source of data might identify the need to include or make a PSS Device Service associated with a specific device. A PSS Device Service might already exist in the FACE Registry. However, not all PSS Devices Services for the same device will provide the same data. Is the Device Service for a specific 429 port? Is it as a 1553 Bus Controller or as a Remote Terminal? Any reuse of a PSS Device Service must be examined closely to ensure it meets the needs of the specific integration.

The base system might identify the need for a menu system that must be modified to support additional capabilities. The software for the user interface will need to be extended as each capability is added. It is possible to develop this software in a completely configurable manner, thus avoiding the need for modification of these components for each integration. The configuration and the system test cases for configurable components will have to derive from these integration requirements.

System Test Cases

Test cases developed for the Independent System and each Independent Capability will be combined into System Test Procedures following the Integration Specification's User Interface and I/O.

Remaining Steps

Once these steps are completed, the rest of the software development process can proceed as normal, developing the designs, software, and lower level tests to the correct higher level artifacts.

The Burden of Extra Documentation

The drive to OSA and development of portable capabilities is not to reduce the cost of the initial development or deployment, but to reduce the cost of development and sustainment of future systems. The program that is developing the initial system should expect some increase in cost due to the needs of the future systems.

We should not, however, burden the initial system with the entire requirements set of the future systems, as it could lead to delays in schedule and unnecessary increase in cost. An adaption of the Agile development method of avoiding gold-plating by only developing to the needs of the next iteration could prove valuable. We should develop the capability to the needs of the current system, but in a portable way. As other systems take on the capability they can expand the functionality. As the capability is expanded to new uses in a flexible and configurable way, it can exponentially increase the number of users that can adopt it without modification.

Independent Capability Document Tree

Generally, an independent capability will be captured as a PCS component. There may be a few capabilities that will include specific PSS Device Services, but this should not be the norm. The data inputs and outputs for an independent capability should be expressed in terms of the data modeled TS Interface.

The artifacts for an Independent Capability should include all artifacts listed in DO-297 for Application Acceptance (DO-297 Section 4.3, including related artifacts in section 4.2) for the required Design Assurance Level.

The artifacts stored with each independent Capability should include all the artifacts needed to support qualification of the software. Future integrations might need to modify or regenerate these artifacts, but in most cases the re-deployment of the capability should not necessitate any modifications.

The artifacts should include all low level test cases, notes on code coverage, and a reusable test harness developed to the TSS interface. This should ease the recreation of test results when the component is ported to new hardware.

When modeling the Independent Capability it is important to keep the model files separate from models for other components within the system. The System Integration Document Tree will link these independent models together to show the full functionality of the integrated system.

The artifacts should also include the system integration test cases that will be developed into System Test Procedures within the integration documentation.

When some of the artifacts are stored with the TRF, the TRF should also maintain the Plans, Standards, CM and QE artifacts.

When we develop the artifacts in this way we also gain the benefit of separating the artifacts needed for FACE Conformance from the rest of the system. This makes the process of providing Verification Evidence cleaner as all documents submitted to the VA will be only those referencing the specific component.

Independent System Document Tree

The independent system document tree begins with the Independent System Specification and includes all hardware in the base system as well as all infrastructure software such as:

- Operating Systems
- Device Drivers
- TSS
- Configurable Graphic Services
- Configurable User Interface
- Other PSS Common Services

Because it represents the basic infrastructure, the independent system is not likely to include any PCS Components, unless they fall into one of the above categories.

The Independent Capability the artifacts for the Independent system should include all artifacts listed in DO-297 for IMA System Acceptance (DO-297 Section 4.4) for the required Design Assurance Level. Some of the software functions should follow the IMA guidance for Application Acceptance (DO-297 Section 4.3) or Module Acceptance (DO-297 Section 4.2).

A reusable test harness may not be possible with some components that are closely tied with the system hardware, but it should be possible to generate one for any PCS, PSS or TSS component.

The models for the Independent System will be more complex than those for the Independent Capabilities. There may be cases in these models where a generic capability is used in place of any specific capabilities for the integration system. Keeping the specifics out of this model will allow for replacement of initial capabilities without requiring modification to these models.

Once again, the artifacts should also include the system integration test cases that will be developed into System Test Procedures within the integration documentation.

The Integration Document Tree

The Integration Document Tree starts with the Integration Specification and includes at a minimum an integration model and the System Test Procedures along with the Plans, Standards, configuration management, and quality assurance records supporting the integration activities.

In addition, any software developed or modified as part of the integration effort will need the artifacts to support the qualification of the software to the appropriate DAL level.

This document tree serves as the highest level document for the system qualification. All capabilities, as well as the hardware and software infrastructure are included into this tree through the pattern of procuring components to meet specifications and standards outside of this document tree.

The System Test Procedures will be developed to use the specific configurations of the components as specified by the documentation in this tree.

Adding or Modifying a Capability

When a new capability is added to the system the Integration Specification and its related artifacts will most likely need to change. Unless the new capability exceeds the capacity of the Independent System, no other documentation would need to change.

If a capability within the system must be modified, the reason for the change should be driven into the artifacts for the modified capability. As capabilities are modified, the changes should include potential refactoring to ensure the software can support maximum portability.

Conclusion

Reduced Integration Time

By developing requirements in an independent manner, programs developing software to FACE, JCA, and other OSA can support maximum portability of the qualification artifacts. Integration time can be vastly reduced through reducing the impacts on artifacts unaffected by the added capabilities. Testing on the target system can be reduced through using the common TSS interface as the interface to automated Independent Capability software test procedures, this maximizes the re-use of the software test cases.

Increased Portability

When a new system makes use of an Independent Capability, the Independent Capability may not include everything that the new system needs. Any requirement for the new system can be added to the requirements of the Independent Capability. The development of these new requirements combined with the old combined with potential refactoring, can lead to a new Independent Capability with greater flexibility.

If capabilities are developed to meet the needs of all systems using the RFA (where the capability requirements are stored), highly adaptable and configurable components can be developed. The likelihood that the Independent Capability can be adopted without change for some future system is greatly increased. Use of this pattern for capturing requirements in the RFA and deriving development as an Independent Capability could be the answer to the projected path where each system tailors a common component to meet single system use.

The more programs subscribing to these principals, the easier it will be to incorporate new capabilities across varying avionics systems.

References

Developer's Handbook for Airworthy Reusable FACE Units of Conformance - Rev 1.4, US Army AMRDEC Software Engineering Directorate, 25 APR 2014

Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations, RTCA, Incorporated, 8 NOV 2005

AC 20-170 – Integrated Modular Avionics Development. Verification, Integration and Approval using RTCA/DO-297 and Technical Standard Order C153, Federal Aviation Administration, 28 OCT 2010

"Open Systems Architecture Contract Guidebook for Program Managers, Version 1.1, May 2013"

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

About the Author

Christopher J. Edwards, PMP has been working in the avionics industry for 20 years, mostly on cockpit systems for military aircraft. In those years he has served in leadership roles in Software, Requirements, System Design, PVI development, Qualification Testing, and Project Management. Mr. Edwards has been the primary author of the FACE Conformance Certification Guide and the Problem Report/Change Request (PR/CR) Process and a contributor to several other documents in both the Technical Working Group (TWG) and Business Working Group (BWG). Mr. Edwards currently serves as a FACE Verification Authority Subject Matter Expert (SME) and is the Lead Systems Engineer for the CMS Project.

About The Open Group FACE™ Consortium

The Open Group Future Airborne Capability Environment (FACE™) Consortium, was formed in 2010 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 FACE Consortium is available at www.opengroup.org/face.

About The Open Group

The Open Group is a global consortium that enables the achievement of business objectives through IT standards. With more than 500 member organizations, The Open Group has a diverse membership that spans all sectors of the IT community – customers, systems and solutions suppliers, tool vendors, integrators, and consultants, as well as academics and researchers – to:

- Capture, understand, and address current and emerging requirements, and establish policies and share best practices
- Facilitate interoperability, develop consensus, and evolve and integrate specifications and open source technologies
- Offer a comprehensive set of services to enhance the operational efficiency of consortia
- Operate the industry's premier certification service

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