Methods for Avionics Systems to Support Third Party Development

Jason Myren
Technical Fellow
Rockwell Collins
Cedar Rapids, IA, USA

Jake Weaver
Principal Systems Engineer
Rockwell Collins
Cedar Rapids, IA, USA

ABSTRACT

Today’s modern aircraft are built with highly integrated avionics solutions that provide pilots enhanced situational awareness and reduced workload. This paper researches how highly integrated avionics can be architected in a way to simplify modification and augmentation of avionics systems with new features developed by third parties (in addition to the original avionics provider). Our objective is to determine the best technical solution(s) to enhancing avionics system architectures to support third party development. We reviewed technical solutions to enable third party integration, including Future Airborne Capability Environment (FACE), ARINC 661, video input, table driven user interfaces, and application co-hosting. We found that a system implementing all of these solutions concurrently offers maximum flexibility, but a system that, at a minimum, implements the FACE™ Technical Standard and ARINC 661 is an efficient way to enhance an avionics system to support Third Party Development.

INTRODUCTION

Over the past several years there has been a trend from airframe vendors to move from procuring federated components (individual control panels and computers) to integrated avionics systems provided by a single vendor. This integration has reduced size, weight and power and increased the usability of the system for the pilots. These latest cockpits are typically made up of larger displays and integrated controls, including touch screens. This level of integration has significantly increased the difficulty in incorporating third party (Original Equipment Manufacturer (OEM) and other vendor) mission specific avionics into the cockpit. This is especially noted in military rotorcraft that frequently have needs beyond a typical fixed wing passenger transport aircraft, which have been the baseline of integrated avionics systems.

This paper will:

- Determine a set of criteria to evaluate the third party expandability of a system
- Create a set of trial cases for modification of the avionics system
- Prioritize the evaluation criteria based on how important they are to each trial case, and then generate an average importance factor.
- Review technical solutions that can enhance an avionics system to accept third party updates
- Rank the ability for technical solutions to meet the criteria
- Draw conclusions about the technical solutions individually and in combination based on their performance against the criteria.
**MAIN BODY**

**Historical Integration of Mission Functions**

In any aircraft there has always been the need over time to add more avionics capability and functionality than the baseline aircraft provides. Adding these functions used to be accomplished via the addition of federated sensors, processors and displays. Over time many aircraft have started using fully integrated avionics systems. This trend was driven by the need to reduce workload. This condition has made it more complex to integrate mission features into the avionics baseline. Initial approaches to this problem varied from integrating new features directly into integrated avionics systems, to developing proprietary interfaces to integrate commercial and military functions. These methods rarely allowed the aircraft OEM or other third party vendors to add mission functionality to the avionics system without involvement of the original avionics system provider.

With a focus on baseline integrated capability rather than extensibility of the avionics system, the efficient growth of the avionics system is limited. Figure 1 defines how we think of a modern avionics system. A typical avionics provider will supply cockpit displays, some type of computing resource and often a suite of sensors and radios. We define third party integration as anything outside this boundary, even if the third party integrated content resides on the avionics processing resource.

![Figure 1. Avionics System Boundaries](image)

**Criteria to Determine Improved Third Party Interface**

We believe that procurement of highly integrated avionics systems that are extensible, provides better life cycle value for mission aircraft than federated designs or integrated systems that are not extensible but offer lower initial cost of procurement. The complexity lies in understanding which areas to focus on. First, this paper will explore the key design criteria (requirements / objectives) we have found necessary to make an integrated avionics system not only **mission capable**, but **easier to expand and enhance** by a third party.

To determine the criteria to measure system expandability, we started by considering the typical broad categories of enhancements that an avionics system will need. The three overarching categories are adding an additional line replaceable unit (LRU), augmenting existing functions, or modifying existing functions. These cases typically require some level of Human Machine Interface (HMI) and functional modification so we looked for criteria to measure the flexibility of a system in these areas.
As we looked at criteria to measure the ease of modification or augmentation of HMI, we focused on three types of integration activities: the ability to make an integrated HMI, the ability to easily evaluate that design, and then simplifying certification complexity. There are a number of criteria that contribute to this (Figure 2), but four were chosen to be used in our evaluation as they seemed to best represent the category without specifying an exact solution (These elements are highlighted in underlined text in the diagram and described below).

Figure 2. Contributing Factors to Creating an Easy to Modify / Augment HMI

1. **Easy to Add to the Existing HMI Scheme / Obtain Uniform Style**: With highly integrated avionics systems it becomes necessary to augment the existing HMI as there is no cockpit space left for additional controls. To maintain low workload, a uniform HMI style is important. We combined these HMI criteria together because of their deep inter-relation.

2. **Supports Rapid Prototyping**: Rapid evaluation of the HMI is important to modern agile development processes. The ability to resolve issues with the additions and modification of an integrated HMI is critical.

3. **Handle/Allow Multiple DAL**: Many mission functions are not civil certified or of high design assurance. The ability to support low design assurance clients without impact to the core system will allow for reuse of the existing military functions.

4. **Helper Libraries with Artifacts**: Helper libraries, and application programming interfaces (APIs) reduce development complexity and increase the speed at which development can be completed.

Next we reviewed criteria that can be used to determine how easily software functionality can be added to the system. This could be the control code for integration of a new LRU or new functional software. Again we used a cause and effect diagram to capture a number of these elements (Figure 3).

We selected 6 key elements from Figure 3 for criteria in our evaluation of solutions.
Figure 3. Contributing Factors to Simplify the Addition of Functionality

1. COTS / Open Interfaces: Having Commercial Off The Shelf (COTS) or open interface standards greatly simplifies the ability to add new functions to the system.

2. Incremental Software Loading: The ability to isolate the mission software load from the avionics load should reduce downtime by reducing complex evaluation of the update, especially time spent on flight test evaluation.

3. Flexible Development Options: The ability to easily add new software can be limited if software porting must be done. If the hardware and platform software environment can support many different development methods, it should be easier to add software to the existing avionics system.

4. Good Data Model / ICD: We expect most additions to require integration with the existing avionics system’s data flows. A good Interface Control Document (ICD) and/or data model is key to this.

5. Clear Understanding of Interactions: Certification requires evaluation of the change, especially to determine the amount of retest required. If the architecture provides a clear understanding of interactions between components it should be easier to limit the amount of retest required.

6. Partitioning / Enclaves: Partitioning and the availability of software enclaves is a clear way to understand and limit the impacts of change.

These previously listed criteria will help identify which technical approaches to improving the third party avionics interfaces will be most beneficial. To do this the criteria will be ranked against each other using modification trial cases based on our real world experience of upgrading avionics systems.

Modification Trial Cases

Trial Case 1 [Multi-function Information Distribution System (MIDS)]:
MIDS is an advanced command, control, communications, computing and intelligence system intended to support key theater functions such as surveillance, identification, air control, weapons engagement coordination and direction for all services.

In this trial case, the MIDS will be integrated on a separate processing computer from the avionics
system. The separate computer will be responsible for processing all MIDS functions. To support this, the avionics system will require spare Input/Output (I/O) to integrate the MIDS. Additionally the MIDS will need to receive aircraft state data from the avionics system.

A MIDS system will require the following HMI features:

- Control Window
- Interactive Menus for MIDS reports
- Display of a layer on the Map
- Crew Alert System (CAS) messages
- Reporting on the integrated Maintenance System

In summary, a MIDS is not highly integrated with the avionics system in the sense that it is replacing functions in a significantly invasive manner. A MIDS mainly augments already existing avionics capabilities.

Trial Case 2 [Very/Ultra High Frequency (V/UHF) Radio Integration]:
In this trial case, a V/UHF comm radio will be added to the avionics radio tuning function. In many cases, V/UHF control code is already existing, so for the purposes of this trial case, the V/UHF control code will be hosted on the avionics system. Spare processing is required along with a partitioned architecture to support hosting new software. This trial case also assumes that a spare bus is available to support integrating the radio.

A V/UHF system will require the following HMI features.

- Control window
- An element on the Primary Flight Display (PFD) to show active frequency
- CAS messages
- Reporting of V/UHF status on the integrated Maintenance System

Trial Case 3 [New Interactive Synoptic for an Electronic Circuit Breaker (ECB) System]:
Addition of an ECB Interactive synoptic typically occurs during the initial design of the aircraft or during initial aircraft development or during a major block upgrade. This interactive synoptic controls all or many of the aircraft circuit breakers. For the purposes of this trial case, the software to control the ECB resides on a separate computer and is integrated with the avionics system via ARINC 429. A typical ECB control function is developed to Design Assurance Level (DAL)-C/B.

The interactive synoptic should display the status of all circuit breakers and provide the flight crew with a means to pull or engage individual breakers. The ECB synoptic should look like it is uniformly part of the avionics system synoptics.

Trial Case 4 [Integration of an Auto Re-Router Function]:
In this trial case an Auto Re-Router function is added to the avionics system. An Auto-Rerouter is typically an embedded software function hosted on the avionics system processing resources. Spare processing is required along with a partitioned architecture to support hosting new software. The auto re-router needs to communicate with the Flight Management System (FMS) and also receive aircraft state data.

The Auto Re-Router will have limited to no HMI. It will display the results of its computations via the existing flight plan capability that is part of the FMS. The Auto Re-Router in this trial case is developed DAL E and is communicating with DAL-C+ Avionics, so it must be integrated in a way that doesn’t cause hazards.

Trial Case 5 [Integration of a Military Surveillance Radar]:
Addition of a military surveillance radar requires an invasive modification of the aircraft structure and wiring. For the purposes of this trial case, the Radar includes a computer that performs control of the radar modes and processing of data acquired by the radar.

The radar will require the following HMI features:

- Control window for moderately complex integrated controls
- Video interface to display radar imagery
- The capability to display the active radar scan area on the map
- The capability to add target data as a map symbology feature
- CAS messages
- Reporting of radar status on the integrated Maintenance System
Trial Case 6 [Replacing the existing FMS performance function with a third party performance function]:
When integrating an avionics system on a platform that has security requirements or unique performance characteristics (such as vertical takeoff and landing capability), the OEM may wish to provide the performance function for the aircraft, rather than providing the data to generate a performance engine to the avionics system provider. In this trial case, the FMS performance capability will be replaced by a third party performance engine running on a separate computer. This type of development is normally performed during initial aircraft development or during a major block upgrade.

Adding a new performance engine requires replacing HMI elements merged into the display with new HMI from the updated perf module. The added HMI is required to have the same look and feel of the avionics system HMI.

Trial Case 7 [Integration of a Mission Digital Map]:
Many military operators require a Digital Map that is more precise with more data than typical civil avionics systems. In this trial case, the Digital Mission Map is running on a separate computer and integrated with the avionics system via a video interface and ARINC 429.

The video interface provides an underlay for integration with the stick map. The avionics system will share lat/long data for the map, along with map center & current map range. The digital map system will provide some non-integrated menuing for map layer/configuration. Finally, the digital map system will report CAS messaging and report computer status to the integrated maintenance system.

Evaluation of Criteria Versus Trial Cases
We evaluated the importance of the criteria versus the trial cases to generate an overall criteria ranking. We present a summary of our ranking for each trial case in Table 1, where a rating of 1 represents this criteria was most important to the trial case and a rating of 10 represents this criteria was least important to the trial case.
### Table 1. Trial Case Ranking*

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Trial Case: 1</th>
<th>Trial Case: 2</th>
<th>Trial Case: 3</th>
<th>Trial Case: 4</th>
<th>Trial Case: 5</th>
<th>Trial Case: 6</th>
<th>Trial Case: 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremental Software Loading</td>
<td>6</td>
<td>7</td>
<td>10</td>
<td>5</td>
<td>9</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>Supports Rapid Prototyping</td>
<td>8</td>
<td>10</td>
<td>3</td>
<td>7</td>
<td>8</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Clear Understanding of Interactions</td>
<td>2</td>
<td>5</td>
<td>9</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Partitioning / Enclaves</td>
<td>10</td>
<td>3</td>
<td>7</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>Handle/Allow Multiple DAL</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>Easy to Add to the Existing HMI Scheme / Obtain Uniform Style</td>
<td>3</td>
<td>9</td>
<td>1</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Good Data Model / ICD</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>COTS / Open Interfaces</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>6</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Flexible Development Options</td>
<td>9</td>
<td>1</td>
<td>8</td>
<td>4</td>
<td>10</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Helper Libraries with Artifacts</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

*Underline for Top Ranked Criteria in a Given Trial Case

We performed detailed justification of the ranking of criteria for each trial case. In summary we found:

**Incremental Software Loading:**
- This typically rated low, as in many of our trial cases, the aircraft hardware modification was going to take time to implement resulting in significant aircraft downtime.

**Supports Rapid Prototyping:**
- This varied from moderately important to less important and reflected how “off the shelf” the trial case proposed the HMI to be.

**Clear Understanding of Interactions:**
- This rated in the middle for most trial cases, depending on the level of interaction between third party developed systems and the baseline avionics system, yet this is an area that should always be considered because it can lead to integration difficulties.

**Partitioning / Enclaves:**
- This varied greatly depending on if the trial case would want to host software on the existing avionics system or not. Since the majority of trial cases did not host software in the existing avionics system, the average ranking of this criteria was lower.

**Handle/Allow Multiple DAL:**
- This rated highly as most of the trial cases were of a lower design assurance and would benefit significantly by being maintained at a low DAL.

**Easy to Add to the Existing HMI Scheme / Obtain Uniform Style:**
- This rated at a maximum both high and low because some trial cases had very limited HMI interactions and other were highly integrated. Overall this rated high because most trial cases involved some HMI.

**Good Data Model / ICD:**
- This rated high, but not as high as HMI. In many cases most difficult aspect was considered...
to be HMI integration and maintaining a low design assurance and thus good data model was ranked below those criteria. Also in a number of cases there was little interaction with the baseline system outside of the HMI.

COTS / Open Interfaces:
- This varied widely based on the trial case and if the software developed for the trial case would need to be hosted in the same computer as the baseline avionics system.

Flexible Development Options
- On average, this ranked lower as the trial cases typically had already developed software and did not benefit as much from this level of capability.

Helper Libraries with Artifacts
- This ranked higher than Data Model / ICD, as having a library and artifacts could reduce ambiguity and then overall effort of software verification. This was ranked relatively high as we expected the libraries would typically be around critical system services required for integration.

After performing the rankings we averaged the ranking results and inverted the result to create an importance factor (subtracting the ranking from 11). Table 2 lists the criteria in order of importance factor:

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Importance Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle/Allow Multiple DAL</td>
<td>7.00</td>
</tr>
<tr>
<td>Easy to Add to the Existing HMI Scheme / Obtain Uniform Style</td>
<td>6.86</td>
</tr>
<tr>
<td>Helper Libraries with Artifacts</td>
<td>6.86</td>
</tr>
<tr>
<td>Good Data Model / ICD</td>
<td>6.14</td>
</tr>
<tr>
<td>COTS / Open Interfaces</td>
<td>6.00</td>
</tr>
<tr>
<td>Clear Understanding of Interactions</td>
<td>5.43</td>
</tr>
<tr>
<td>Supports Rapid Prototyping</td>
<td>4.71</td>
</tr>
<tr>
<td>Partitioning / Enclaves</td>
<td>4.43</td>
</tr>
<tr>
<td>Flexible Development Options</td>
<td>3.86</td>
</tr>
<tr>
<td>Incremental Software Loading</td>
<td>3.71</td>
</tr>
</tbody>
</table>

Technical Approach to Improving Third Party Interface

This section of the paper will review a number of techniques that have the ability to address the criteria identified in the previous section. These techniques will be based on methods researched and used at Rockwell Collins and will include solutions for adding HMI and software functions. The technical approaches we will review are:

- **HMI Focused**
  - ARINC 661, which allows a composite HMI built by many client applications
  - Video, both displayed video and interactive video

- **Functional Software Focused**
  - FACE Interfaces, which is utilized by portable software components, such as a Flight Management System (among other functions)
  - Table Driven User Interfaces, for functions such as user defined Crew Alerting System
  - Co-hosting supplier and OEM / third party software on the same computing platform

**ARINC 661**

The ARINC 661 specification seeks to standardize the definition of a Cockpit Display System (CDS) and the communication between the CDS and the User Applications (UA) which manage aircraft avionics functions (Ref. [6]). The Graphical User Interface (GUI) definition is completely defined in Binary Definition Files (BDF).

The ARINC 661 specification allows multiple software applications to provide formats to the CDS in a way that allows each format to be independently managed. The CDS can then generate a seamless merged display by layering the independent application data across any connected display. Figure 4 provides an example of the ARINC 661 concept of layering application data to create a seamless display.
The advantage of allowing multiple software applications to render different layers of a single display provides the following benefits for third party integration:

- Cleanly defined standard that is commonly used throughout the industry
- Widely available tools for development of graphics
- Standard styleset and common widgets that promote uniform look and feel.
- Separation of DAL allowing multiple applications of different levels to draw to the same screen
- Enables rapid prototyping for easy validation

Our approach to enabling integration via ARINC 661 is to provide an ARINC 661 toolkit. This toolkit simplifies the development process for the end user. The toolkit provides simple Application Programming Interface (API) initialization and seeks to minimize what the developer must know regarding the underlying communication and ARINC 661 protocols, by handling the low-level details of the communication protocol automatically. This toolkit takes advantage of common industry tools and strives to enable a third party developer to begin drawing graphics to a CDS within 1 hour of setup.

Allowing a third party to use ARINC 661 to draw to a commercial CDS requires collaboration between the OEM and the third party. This collaboration can take different forms. For example, as we worked with General Atomics on the SkyGuardian, our developers were heavily involved in the human machine concepts that were eventually fielded as part of that ground station. On the opposite end of the spectrum, as we work with various OEMs, they bring their own style and generally do the majority of their specific development with minimal input from our teams.

Avionics providers can work to address this collaboration by provisioning systems with features that would allow for typical third party customization. Examples of this include:

- Allocating connector widgets and layers for the addition of new map features
- Allocating layers to provision for the inclusion of new navigation and communication radios
- Allocating connector widgets to add new alerts driven from third party systems
- Allocating connector widgets in menu lists for the addition of new features

**Video**

A simple video interface can provide significant flexibility to add features to an avionics system. Standard protocols such as ARINC 818, Digital Video Interface (DVI) and SMPTE-292 provide a low cost, low impact method to add features to an avionics system. Video can be interactive or simply for display purposes.

The main drawback of video is that it offers very little flexibility in terms of matching the overall look and feel of the avionics system. Additionally, interactive video can sometimes lead to difficulties with cursor interaction. For example, determining cursor control and position when going from a video drawn graphic to an ARINC 661 drawn graphic can cause difficulties in integration. While not insurmountable, this is a challenge that must be understood when integrating interactive video.

**FACE Open System Interfaces**

The Open Group® FACE Consortium has created technical (Ref. [2]) and business guidance (Ref. [3]) focused on a common architecture for use in avionics systems.

The FACE Technical Standard makes it possible to not only create an open environment to run applications, but specifies the creation of conformant applications to use in the architecture. It also encourages building a product shelf of conformant
applications to choose from. Building an avionics system from a product shelf naturally lends itself to building systems that include third party applications.

Using the FACE Technical Standard helps build a system that is designed for expandability. The Standard provides three key tenets that are valuable to allowing third party expandability of a system:
- Open Interfaces
- Layered approach of software construction, prevent one big mass of software
- Well documented interfaces

First, the FACE Technical Standard is an open standard available via the Open Group. All developers correctly using it can create software they know will work with a FACE conformant computing environment. Likewise a system integrator knows that FACE conformant applications will work with their platform. This strategy is not built on trust, but verification. Each FACE Unit of Conformance can earn a certificate of conformance after submitting verification results to an independent Verification Authority.

Second the FACE Technical Standard requires separation of the implementation into segments, which have defined interface rules. These segments are:
- Portable Components Segment (PCS)
- Transport Services Segment (TSS)
- Platform Specific Services Segment (PSSS)
- I/O Services Segment (IOSS)
- Operating System Segment (OSS).

Key APIs defined by the FACE Technical Standard allow components to interface with components residing in other FACE segments, as well as to components residing external to the FACE architecture. The key APIs are the Operating System (OS) interface, Transport Services (TS) interface, and Input Output (IO) interface as seen in Figure 5. This design discourages creation of a single monolithic application, and forces separation of the I/O, business logic and human machine interface.

Third the FACE Technical Standard requires that all data transferred between components be well documented in data models. Connections between the PCS, PSSS and IOSS cannot simply open channels but instead must clearly define each message. While the intention is to make it possible for the integrator to bring together off the shelf pieces to create their system, this has significant advantage to create a system that allows third party expandability.

With the preceding tenants in mind, we see two easy ways to expand a system built with FACE components after initial delivery with third party software:
- Addition of functional clients
- Addition of new HMI elements

Addition of Functional Clients

The segmented approach of the FACE architecture encourages a design that separates business logic into Units of Portability running in the PCS segment. Typically, these business logic segments need to support interaction with the human machine interface in the PSSS layer. While not strictly required, it is likely a PCS Unit of Portability will support requests from multiple PSSS or PCS clients. This means it will be relatively easy to add additional software that interacts with these functions.
A conceptualization of this capability is the creation of a basic system with a HMI to allow the user to load flight plans. The system is later updated with FACE PCS components that are used to create mission flight plan segments. The new PCS components use the same interface as used by the baseline HMI (Figure 6).

In the example of the MFMS-1000, it is only a PCS function. That makes it possible to build a system with the MFMS-1000 and integrate a new HMI in whichever way best suits the architecture, which can be in a way that best allows third party additions.

Thus imposing the FACE segmented approach on an architecture immediately and naturally opens the ability to break down barriers that prevent third party expandability in the system. On the other hand FACE does not require use of its allowed user interfaces (ARINC 661 or OpenGL) in a way that allows expandability.

Table Based User Interfaces

To suit the varying skills of third party developers, as well as meet the needs for varying levels of complexity, there are several different user interfaces, from simple to more complex, that should be considered. Examples of non-software capabilities we’ve explored and implemented are:

- Strapping: Configuring the system via a physical means, such as a configuration screen available to the installer, license keys or physical hardware strapping (e.g. configuring for 2 Global Position System (GPS) sensors or 3 GPS sensors).
- Non-software solutions: Capabilities include configuration tables for various aircraft systems, such as FMS performance, Crew Alert messaging or Maintenance Reporting.

Strapping provides a relatively straight-forward means of allowing a third party to configure an avionics system. On our Pro Line Fusion® system, for example, we have two means that allows a third party to configure the system. The first is a simple User Configuration format. This format is a maintenance feature that allows for selection of numbers of LRUs. For example, enabling a 3rd Very High Frequency (VHF) radio or a 2nd Distance Measuring Equipment (DME) radio. When a maintainer enables an option, this option is loaded into the software so that when the system is restarted, it configures appropriately.

An additional method of configuration that we’ve employed is a type of strapping using License Keys.
In our Pro Line Fusion® system, we use license keys to enable certain features that can be aircraft or customer specific. The system includes all possible license keys, and when a new license key is enabled, this option is loaded into the software so that when the system is restarted, it configures appropriately.

The next level of complexity for adding capability to an existing avionics system involves table driven modification. In our experience, three common areas that can be addressed by table driven modifications are visual crew alert messages, audio alert messages and maintenance recording/reporting. The ability to add these features is critical to third party development because any time a new LRU is added to a system, it will generally require some Crew Alert System (CAS) messages and maintenance reporting. Additionally, depending on the type of system, aural alerting may be required.

CAS provides the primary means to notify the flight crew of aircraft issues. CAS messages generally come in the form of Warnings, Cautions and Advisories. There is generally at least 1 CAS alert per integrated system. Because of this, a third party modifying the system needs to be able to add CAS messages to the avionics system.

A method we’ve successfully employed is using tables to drive CAS messages. In this implementation, CAS software handles inputs, performs computations on inputs, and transmits results of computations as outputs. The CAS software contains an interpreter to read the tables. The CAS tables define input conditions that will trigger CAS messages and the resulting CAS output message.

The primary method we’ve deployed is splitting the CAS table between avionics provider CAS messages and third party CAS messages. The CAS engine reads multiple tables, allowing a third party developer to simply define their own table.

Similar to CAS messaging, we’ve successfully fielded table driven logic for the generation of aural alerts. Additionally, our aural alerting systems use simple .wav files for the actual audio file, which is a standard format most audio generation tools can create.

Maintenance reporting has the potential to be more difficult than CAS alerting or Audio alerting due to the complexity of many modern aircraft maintenance systems. However, similar methods can be applied.

By splitting the maintenance system into an application layer and a table layer, it is possible to support maintenance reporting via table driven methods, allowing 3rd parties a relatively simple way to add maintenance data for added systems.

Co-Hosting Supplier & Third Party SW on Computing Platform

The previous technical approaches focus on interfaces to the baseline avionics system and do not emphasize where the third party features are hosted. Thus it is important to look at ways to host third party software that do not require additional processing modules. Many times it will not be acceptable to add more cost, weight and power of a new processing unit. One way to solve this problem is to provide space on the baseline avionics to host third party applications. We refer to hosting third party software on vendor hardware as co-hosting.

Co-hosting software can have initial roadblocks. Typically the hardware platform and its software interface layers can have vendor specific IP that the first party isn’t prepared to share, but is necessary for the third party to use the hardware platform. Even when this is not an issue, the third party may still be faced with a significant effort to port software to a different processor due to such factor such as Operating System (OS), I/O interface or differences in configuration. These roadblocks build until co-hosting seems like an impossibility.

We have experience with two methods to address this. First is to have a standard platform that offers a uniform set of known services. Examples include ARINC 653, POSIX, and the FACE Technical Standard. This does require the standard have some broad acceptance to be useful, as the platform and the third party need to both be using the same solution, and the software has to been constructed with this type of portability in mind (especially if it can be
flexible between ARINC 653 and POSIX interfaces for example).

A second way to address the co-hosting issue is the use of virtualization. Virtualization allows creation of a virtualized hardware environment with its own OS, I/O and middleware concept. Then the third party can have the environment most compatible with their software design, without impact to the first party. This is the equivalent of adding new computing for the third party without the cost of hardware installation. Virtualization has become very common in the commercial Information Technology (IT) world, with heavy use in the web services community (Amazon Web Services for example rents out virtualized machines for users to develop to).

Rockwell Collins has utilized hypervisor-based virtualization to enhance the partitioning between functions for a flight critical system. By significantly eliminating any coupling between the first party environment and the third party environment it is much easier to host third party software.

A key element of a virtualization architecture is the inclusion of reserve available for allocation of guest OS. This could be done by reserving general spare processing for a guest OS, or even reserving an entire core in a modern multi-core processors (Ref. [8]). As an example, a multi-core processor primarily using Lynx Software Technology LynxOS-178 Guest OS could have a core of the processor reserved as spare. Then if an existing missionization function were developed for WindRiver VXWorks, it would be possible to run that Guest OS in the baseline system’s reserved processor core. In addition to running other real time OSes it is possible to run a Linux distribution for low design assurance environment (Ref. [9]) without adding additional processors.

Technical Approach Evaluation

In this section we will evaluate the technical approaches against the criteria defined earlier. We will start by evaluating each major technical approach and then look at how those technical approaches rank when combined.

To evaluate the technical approaches against the criteria we rated the ability for the technical approach to support the criteria. If the technical approach clearly supported the function it was ranked as “yes”, if it could support the criteria depending on how the approach was used it was ranked “maybe”, and finally ranked “no” if there wasn’t a significant way the criteria could support the approach. A score was assigned to the ability:

- Yes: 2 points
- Maybe: 1 points
- No: 0 points

Tables 3 is our assessment of the technical approach to meet the criteria.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>A661</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremental Software Loading</td>
<td>No</td>
<td>Does not affect how software is loaded, just HMI integration.</td>
</tr>
<tr>
<td>Supports Rapid Prototyping</td>
<td>Yes</td>
<td>ARINC 661 is a widget based design allowing quick refactoring of the HMI. Many ARINC 661 tool providers include rapid prototyping capabilities.</td>
</tr>
<tr>
<td>Clear Understanding of Interactions</td>
<td>Yes</td>
<td>ARINC 661 only involves HMI. It is able to clarify HMI interactions through use of connector widgets and containers.</td>
</tr>
<tr>
<td>Partitioning / Enclaves</td>
<td>Maybe</td>
<td>ARINC 661 use of connector widgets and containers allows a clearly defined enclave to be created, but is only for HMI.</td>
</tr>
<tr>
<td>Criteria</td>
<td>Video</td>
<td>Justification</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Handle/Allow Multiple DAL</td>
<td>Yes</td>
<td>ARINC 661 in most implementations has been used to build a composite HMI from multiple DAL.</td>
</tr>
<tr>
<td>Easy to Add to the Existing HMI Scheme / Obtain Uniform Style</td>
<td>Yes</td>
<td>As a widget based design, the HMI style is held in the avionic system software, not in the client applications, simplifying uniformity.</td>
</tr>
<tr>
<td>Good Data Model / ICD</td>
<td>Maybe</td>
<td>ARINC 661 has a well-defined data model managed by committee but only covers graphics capabilities.</td>
</tr>
<tr>
<td>COTS / Open Interfaces</td>
<td>No</td>
<td>ARINC 661 is for HMI and does not address open interfaces in reference to adding additional LRUs.</td>
</tr>
<tr>
<td>Flexible Development Options</td>
<td>No</td>
<td>ARINC 661 is an HMI method and does not provide for software development specifications.</td>
</tr>
<tr>
<td>Helper Libraries with Artifacts</td>
<td>Yes</td>
<td>The ARINC 661 interface can easily be turned into a library that simplifies building client code.</td>
</tr>
</tbody>
</table>

**FACE**

<table>
<thead>
<tr>
<th>Criteria</th>
<th>FACE</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremental Software Loading</td>
<td>Maybe</td>
<td>The FACE Technical Standard allows significant flexibility including safety critical and non-safety critical designs. It does not require the architect to use these different domains, so it is up to the architect to leverage this feature.</td>
</tr>
<tr>
<td>Supports Rapid Prototyping</td>
<td>No</td>
<td>The Standard concerns software architecture not HMI, and does</td>
</tr>
<tr>
<td>Criteria</td>
<td>Table Driven</td>
<td>Justification</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Clear Understanding of Interactions</td>
<td>Yes</td>
<td>A data model is required and its use is enforced between system segments. The Standard also limits what can be done in a layer.</td>
</tr>
<tr>
<td>Partitioning / Enclaves</td>
<td>Yes</td>
<td>The FACE Technical Standard supports an OS segment with partitioning, and requires software delivered in components.</td>
</tr>
<tr>
<td>Handle/Allow Multiple DAL</td>
<td>Maybe</td>
<td>The Standard supports an OS segment with partitioning and safety critical design, but does not require the architect to use it.</td>
</tr>
<tr>
<td>Easy to Add to the Existing HMI Scheme / Obtain Uniform Style</td>
<td>Maybe</td>
<td>Use of ARINC 661 or OpenGL is supported. OpenGL itself does not help maintain an integrated HMI scheme.</td>
</tr>
<tr>
<td>Good Data Model / ICD</td>
<td>Yes</td>
<td>The FACE Technical Standard requires a data model and does a verification activity on it.</td>
</tr>
<tr>
<td>COTS / Open Interfaces</td>
<td>Yes</td>
<td>The IO Services Segment helps to ease the addition of new hardware into the existing data management scheme.</td>
</tr>
<tr>
<td>Flexible Development Options</td>
<td>Maybe</td>
<td>Many OS layer options are allowed but flexible deployment depends on the baseline system configuration.</td>
</tr>
<tr>
<td>Helper Libraries with Artifacts</td>
<td>Yes</td>
<td>There is the opportunity for many helper libraries. As an example the TSS library.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Co-Host</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremental Software Loading</td>
<td>Yes</td>
<td>Strapping or table driven updates, doesn't affect how software is loaded.</td>
</tr>
<tr>
<td>Supports Rapid Prototyping</td>
<td>Yes</td>
<td>Strapping or table driven updates allows quick modification and verification of stakeholder needs.</td>
</tr>
<tr>
<td>Clear Understanding of Interactions</td>
<td>Yes</td>
<td>Strapping or table driven updates allows easy modification to the system that is simple for a developer to understand.</td>
</tr>
<tr>
<td>Partitioning / Enclaves</td>
<td>Yes</td>
<td>Does not require partitioning/enclaves. Relies on configuration tables and/or hardware strapping.</td>
</tr>
<tr>
<td>Handle/Allow Multiple DAL</td>
<td>Yes</td>
<td>Does not affect DAL. Loading of new tables/files to be read by existing DAL applications.</td>
</tr>
<tr>
<td>Easy to Add to the Existing HMI Scheme / Obtain Uniform Style</td>
<td>No</td>
<td>This method is not an HMI tool, but a method to expand on what existing applications already do.</td>
</tr>
<tr>
<td>Good Data Model / ICD</td>
<td>Maybe</td>
<td>Avionics system provider specific, meaning complexity will vary from avionics system to avionics system.</td>
</tr>
<tr>
<td>COTS / Open Interfaces</td>
<td>No</td>
<td>Avionics system provider specific. No standard exists.</td>
</tr>
<tr>
<td>Flexible Development Options</td>
<td>No</td>
<td>Avionics system provider specific. Requires exact definition to match existing avionics system needs/formatting.</td>
</tr>
<tr>
<td>Helper Libraries with Artifacts</td>
<td>Yes</td>
<td>Helper libraries with artifacts provide proper training and examples to follow this approach.</td>
</tr>
<tr>
<td>Criteria</td>
<td>Importance Factor</td>
<td>A661</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-------------------</td>
<td>------</td>
</tr>
<tr>
<td>Incremental Software Loading</td>
<td>3.71</td>
<td>No</td>
</tr>
<tr>
<td>Supports Rapid Prototyping</td>
<td>4.71</td>
<td>Yes</td>
</tr>
<tr>
<td>Clear Understanding of Interactions</td>
<td>5.43</td>
<td>Yes</td>
</tr>
<tr>
<td>Partitioning / Enclaves</td>
<td>4.43</td>
<td>Maybe</td>
</tr>
<tr>
<td>Handle/Allow Multiple DAL</td>
<td>7.00</td>
<td>Yes</td>
</tr>
<tr>
<td>Easy to Add to the Existing HMI Scheme / Obtain Uniform Style</td>
<td>6.86</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Each criteria previously had an importance factor computed. We multiplied the importance factor by the ability of the technical approach to meet that criteria generating an individual rating. Then we summed all the individual ratings for that technical approach. This generated a total rating for that technical approach summarized in Table 4.
These technical approaches are not mutually exclusive. Thus a combination of technical approaches could increase the benefits if applied to an architecture. To evaluate this we generated all possible combinations of the technical approaches. When we combined approaches we assessed the ability to support the criteria based on the best rating of the supporting technical approaches. As an example if ARINC 661 contributed a “maybe” score and the FACE Technical Standard contributed a “yes” score, then the result for an architecture using both standards was “yes”.

It was not surprising that by combining all five solutions results in the highest identical score. Our objective is instead to determine which combinations are most efficient. Thus we computed an efficiency score based on the total score divided by the number of technical approaches used in the given solution. Table 5 shows this sorted list of capabilities:

<table>
<thead>
<tr>
<th>Solution</th>
<th>Total Rating</th>
<th>Number of Solutions Used</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>FACE Technical Standard</td>
<td>79.1</td>
<td>1</td>
<td>79.1</td>
</tr>
<tr>
<td>A661</td>
<td>72.3</td>
<td>1</td>
<td>72.3</td>
</tr>
<tr>
<td>TableDriven</td>
<td>70.4</td>
<td>1</td>
<td>70.4</td>
</tr>
<tr>
<td>Video</td>
<td>62.6</td>
<td>1</td>
<td>62.6</td>
</tr>
<tr>
<td>A661 FACE Technical Standard</td>
<td>102.4</td>
<td>2</td>
<td>51.2</td>
</tr>
<tr>
<td>Video FACE Technical Standard</td>
<td>99.3</td>
<td>2</td>
<td>49.6</td>
</tr>
<tr>
<td>FACE Technical Standard TableDriven</td>
<td>99.3</td>
<td>2</td>
<td>49.6</td>
</tr>
<tr>
<td>CoHost</td>
<td>48.7</td>
<td>1</td>
<td>48.7</td>
</tr>
<tr>
<td>A661 Video</td>
<td>96.1</td>
<td>2</td>
<td>48.1</td>
</tr>
<tr>
<td>A661 CoHost</td>
<td>91.9</td>
<td>2</td>
<td>45.9</td>
</tr>
<tr>
<td>FACE Technical Standard CoHost</td>
<td>86.7</td>
<td>2</td>
<td>43.4</td>
</tr>
<tr>
<td>A661 TableDriven</td>
<td>84.1</td>
<td>2</td>
<td>42.1</td>
</tr>
<tr>
<td>Video TableDriven</td>
<td>82.4</td>
<td>2</td>
<td>41.2</td>
</tr>
<tr>
<td>TableDriven CoHost</td>
<td>78.1</td>
<td>2</td>
<td>39.1</td>
</tr>
</tbody>
</table>
CONCLUSIONS

It is notable from the evaluation that ARINC 661, the FACE Technical Standard and Table Driven User Interfaces scored very high and similarly above 70 points. Each of these individual solutions provides significant capability to support third party development and evolution of the system. Because of this, they will rate higher than any combination (a perfect combination of 2 solutions = 110 and thus would have an efficiency rating of 55, lower than the single solutions). Nonetheless, we feel the results for the ten solution pairings deserve additional attention (Table 6).
The top three solutions all combine with the FACE Technical Standard that is the highest rated stand-alone technical approach. This makes sense since the FACE Technical Standard has the broadest application across architecture tenants. The highest ranked combination is the FACE Technical Standard with ARINC 661. By enforcing the choice of ARINC 661 the combination brings better third party display access.

Second ranked is the FACE Technical Standard plus Video, as even if a relatively closed graphical rendering method is chosen, having the easy to integrate Video solution will allow for HMI expandability. We believe this display access is critical, as in highly integrated cockpits there is little additional space for federated HMI solutions.

Third ranked is the FACE Technical Standard plus Table Driven User Interfaces. While this would seem to be the combination of architectural solutions not focused on HMI, Table Driven User Interfaces, as described in this paper, also provides access to the system HMI through its configuration tables.

The fourth ranked combination of ARINC 661 and Video being as beneficial as it was is surprising to us, as they are both HMI standards. As we thought through this, we realized this combination provides benefits in a system where third party capabilities are integrated on a separate processing resource. In these scenarios, the focus of integration with the avionics system is HMI, as such ARINC 661 and Video provides strong advantages. Retaining video as a simple interface in addition to the more complex ARINC 661 methods provides significant HMI flexibility across a wide variety of third party HMI integrations.

Outside of pairing technical approaches, a system design applying all five of the technical approaches described in this paper still offers the most flexibility. Fundamentally the FACE Technical Standard is a good starting point and then as indicated earlier selecting ARINC 661 and Video addresses HMI flexibility. In addition Table Driven User Interfaces and Co-Hosting help ensure the platform is better able to host logic upgrades in addition to HMI upgrades easily. While supporting the ability to extend HMI was more important than the need to limit the addition of processors and LRUs to the aircraft, when SWAP is considered, a system better able to address the addition of functions without the addition of hardware is beneficial, especially as more and more software functions become available on the “app shelf.”

Overall, the resulting analysis provides an indication of an avionics system design, and its adaptability to the chosen use cases. It is up to the OEM to determine which use cases are the most applicable to the aircraft and thus optimize the requirements that affect the selection of the avionics.

FACE™ is a trademark of The Open Group in the United States and other countries.
REFERENCES


2“Technical Standard for Future Airborne Capability Environment (FACE), Edition 2.1”: Published by The Open Group, May 2014

3“FACE™ Business Guide, Version 1.1”: Published by The Open Group, December 2011


6“ARINC Specification 661-6: Cockpit Display System Interfaces to User Systems”: Published by ARINC, April 2016

