Common Symbology Approach Using Configurable Core User Applications

Stategies for increasing commonality within platforms using the FACE Aproach.

The Open Group FACE™ Army TIM Paper by:

Christopher J. Edwards, Systems Engineering Lead, CMS Team

Steven P. Price, Software Engineer, CMS Team/FACE TSS SC Lead

May 26, 2021

Table of Contents

Executive Summary

The use of configurable core components provides advantages in airworthiness qualification and reducing the effort to add additional capabilities into an existing system. Typically, new capabilities added to a system will have user interface impact related to its control and display.

When it comes to user interface design, it is desirable that the platform has consistent control interfaces and display symbology across all capabilities. A set of configurable core components, such as a menu system, can provide a common integrated look-and-feel to a user. These common components can be expanded to cover other aspects of the presentation of data from new capabilities.

Increasing the number of configurable core components to include common graphical representations of numeric values distributed in the system can greatly reduce the effort to display data from newly added capabilities to the aircraft.

Language of ARINC-661 and User Interfaces

The RIF team has been developing systems following an evolving Future Airborne Capability Environment (FACE) Technical Standard. In tandem, several papers were authored by this team which describe architectural approaches. To reduce cycle times fielding new or upgraded capabilities, systems should employ strategies to reduce airworthiness impacts of new capabilities. The ability to bring new capability to the warfighter in a timely manner has been a principal focus of this team.

Core and Hosted Capabilities

The definition of a Core System and its Hosted Capabilities was first published for a FACE™ TIM in 2017 (Edwards, Price, & Mooradian, October, 2017). Use of Configurable Core Components allows for efficiencies in the qualification of systems as new Hosted Capabilities are added. One of the best examples of a Configurable Core Component is the Menu System (Price & Edwards, 2017).

ARINC-661

The ARINC-661 Standard (AEEC - Engineering Standards for Aircraft Systems, 2019) defines a means for displaying user interfaces in cockpit systems. Under this standard a Cockpit Display System (CDS) operates as a Configurable Core Component. Hosted Capabilities are added to the system as User Applications (UAs). Configuration of the CDS is through a set of Definition Files (DFs) associated with each UA that are loaded into the CDS. Widgets are defined in the ARINC-661 standard as the basic building blocks of a user interface. The DF essentially defines a set of Widgets that a UA will use. ARINC-661 is recommended for use in the FACE Technical Standard Edition 2.1 (FACE Consortium, 24 Jun 2014) Edition 3.0 (FACE Consortium, December 2017), and later.

One pattern expressed in ARINC 661is that the CDS drives look and feel while the UAs provide the function. The goal is to let the platform decide interface specifics for things like specific colors and input devices; and to build the UAs so these interface choices are abstracted. Code reuse in graphics leads to a more consistent look and feel across the integration. Modern user interfaces from Windows to Android include the ability for the implementation of the user interface to be customized by the user. In avionics that customization is reflected in the specific use of the particular user interface and the mode of the system. Driving interface choices to the common core simplifies the addition of new capabilities while keeping consistency of the look and feel for capabilities installed on the platform.

Common Symbology Approach Using Configurable Core Symbology

Model View Controller

Application of the widely accepted Model View Controller (MVC) design pattern greatly aids in the development of portable user interface software.

The MVC design pattern divides applications with user interfaces into three domains:

1. a model that represents the business logic exclusive of any rendering or control,
2. a view that renders the image to the user, and
3. a controller that represents user input systems for interaction with the model.

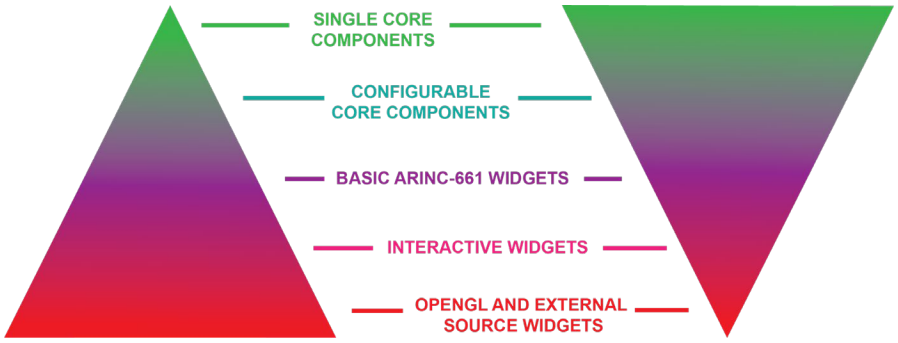
The application of a configurable core system would suggest the view and controller aspects should be handled through configuration as much as possible.

Considerations for Configurable Core View and Controllers

User Interface Choices that Impact Effort and Commonality

When considering the addition of a new capability, or even the modification of an existing one, there is a set of questions that should be asked. When asked in this given order, the first question that can be answered in the positive is the likely approach for adding the capability with minimum effort and impact.

- Is the new capability already represented by a single core component?
- Can the new capability be represented by configurable core components?
- Can the new capability be rendered by basic ARINC 661 widgets?
- Can the new capability be rendered by interactive ARINC 661 widgets?
- Can the new capability be rendered through OpenGL and External Source?



Single Core Components directly tie new capability functions to a clear core capability such as the system status page, an alerting system, and/or a menu system.

Configurable Core Components (the subject of this paper) represent a class of user interface functions that uses capabilities to represent more complex renderings or user interactions. These core components are built to match the style and user interface of the aircraft system while presenting a more generic view/controller interface to the new capabilities.

Basic ARINC-661 widgets provide an option for having the CDS still establish the style of the display, but places the use of the ARINC-661 communications upon a UA associated with the new capability.

Interactive Widgets provide a CDS determined look-and-feel, but the interface is mostly determined by the new capability. This potentially causes disconnect in commonality between multiple integrated capabilities and could lead to additional capability specific code generation.

OpenGL and External Source Widgets drives the least commonality and the most capability specific rendering logic.

The capabilities of the core system will dictate what is available for the new capabilities to use. Some hosted capabilities will be forced into the OpenGL route for display of complex images, but control and status

functions related to the hosted capability may be rendered using configurable core capabilities. This hybrid approach will improve the cohesive nature of the user interface.

Validity, Staleness, and General Availability

The format of a valid input and the indication of a stale or invalid input should be a consistent system wide decision. Hosted Applications that make choices on the way that an invalid or out-of-range value is displayed may need to be altered or recompiled when they are ported to a new system. Handling these displays in a common readout application allows for an easier configuration of system wide format decisions

Using the Transport Interface and Data Transforms

As with the Menu System and a System Status function, that ability to transform values from one Hosted Capability modeled element to a Core Component modeled element is a key aspect of having configurable Core Components (Edwards, Price, & Tanner, Transformation Capabilities in Configurable Common Services, 2018). Numeric data along with its status is often already on the transport, modeled to the observable and measurement matching the capability need. Transforming the numbers to a "Numeric Readout Value" can often be a simple matter of connecting the generic display UA to the correct types.

Configuration of the UA

The UA Configuration should take into account the information related to how the values from the transport should be displayed. Some of the information in this configuration may include the mapping of Style (color, font size, etc) to input value ranges or Boolean input values.

Configuration of the CDS

The Cockpit Display System (CDS) configuration will include at least one Definition File (DF) for the configurable core components. The addition of new graphical representations will likely call for modification or addition of new DFs. This is expected and if the system uses an approach to establish separate configurations for mixed criticality, it should not adversely affect the qualification of the system. (Edwards C. J., 2019).

Configurable Core User Applications (UAs)

If a configurable core component is already available, new capabilities can be added to the user interface without addition of user interface software; the configurable core component is simply configured for the new capability.

Example: An alert system on the aircraft can relay alerts in the form of visuals, sounds and haptic feedback. This system can be constructed to use a configuration file, relying on the Transport Services Segment (TSS) implementation to route the appropriate messages to the system. New alerts are added though a configuration and TSS routing update without impact to the bulk of the qualification artifacts.

The goal with configurable core UAs is to abstract away the bulk of how to present data to a user from the application and/or processing of that data. This can be accomplished by creating a set of UAs that take in generic data from the transport. The presentation of the data to the user then becomes a matter of configuration parameters and the ARINC-661 DF. A small distinct set of configurable core UAs would receive generic data and be configurable to allow the presentation of that same data in a manner consistent with the platform. These configurable core UAs could also have some generic interactions defined.

Use of User Defined UAs in the Context of ARINC-661

Most ARINC-661 development environment tools provide a means to develop user defined widgets and extend the ARINC 661 widget list. While this could make adding new capabilities easier, it does pose some problems. User defined widgets create a custom graphic server and application dependency. That dependency would mean both the graphic server and applications in which it interacts are less portable. It creates concerns even when updating to a new version of the same graphic server. Implementing user defined widgets does not enforce abstraction of how the data is displayed from the application.

Common Interface Sets

Many capabilities present numerical data, often with multiple values related to each other. Integrated platforms may use a tape or dial style for displaying these numeric values; often these accompany a digital readout of the value. Having a common means of displaying a number would allow for rapid addition of these capabilities while presenting the values in a consistent means with the rest of the system.

Many capabilities call for a numeric input, such as a frequency, range, or bearing. Interfaces for numeric inputs might include rotating dials for faster entry. The existence or lack of a keypad is another factor that core capabilities should abstract from the hosted capabilities.

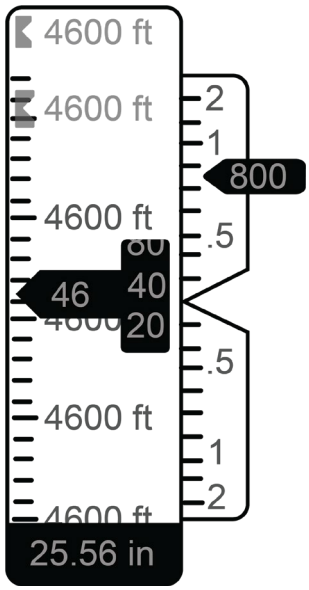
The display of text data is in some ways similar to the numeric values, but often the formation of a text string will call for a Hosted Capability to format that string for consumption by the user. The formatting becomes part of the user interface and the Hosted Capability; but the implementation of the look and feel can still be handled by the configurable Core Capabilities.

Other capabilities call for the selection of a number of options, often these options are of a limited set known at configuration time. Sometimes these options are only presented during the mission, such as the messages received on Blue Force Tracker.

Numeric Representations

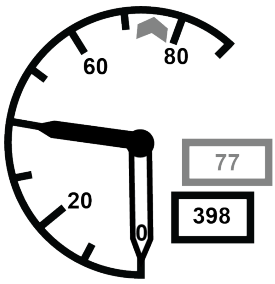
One of the most common things a new capability might bring to an aircraft is a numeric value that simply needs to be displayed, perhaps combined with other numeric values in a cohesive manner. One aircraft platform might favor a dial representation with bugs and limit bands, while another might favor a tape display. Often the manner that a simple numeric readout is displayed may be highly influenced by the style of the overall presentation.

Additional capabilities may be adding a new numeric value that is closely related to existing values within the aircraft. This could include adding a calculated altitude limit to avoid an active radar site, or placing a calculated limit on the radar altitude indicator for a sling load.



Other capabilities may bring their own set of related values that should be displayed in a manner familiar to the style of the existing platform, and could be implemented by reuse of the same code providing the current user interface.

Needles, dials, and even some readouts will have ranges of acceptable values that may affect the display of the data. These ranges may dictate colors of the display. On tape and dial readouts they may represent ranges where the needle deflection varies to provide more detail over tighter ranges of numbers. In the Vertical Speed display to the right the range from +1 to -1 shows more detail than the areas outside of this detailed area. These ranges have to be configured with boundaries on the input values as well as the translated boundaries of the display.



Configuration of the UA

The configuration parameters to consider as part of the generic numeric UA include the values specified in Table 1: Configuration of a Numeric Readout UA. In addition to the items listed, the UA will need to include linkage to the Widget IDs generated in the DF. Linkages can be through a predetermined pattern, use of a base configuration identifier for the widgets in a set, or even through a configuration including all identifiers for all related widgets.

Table 1: Configuration of a Numeric Readout UA

Configuration Catagory	Parameters
Needles	Number, Color, Style
Bugs	Number, Color, Style
Limit Indicators	Number, Color, Style, Effect on Style
Zone Ranges	Number, Input Ranges, Display Range
Digital Readouts	Number, Format, Scrolling Digits, Unit Measure Display

Common Symbology Approach Using Configurable Core Symbology

Modification of the DF

While it is possible to implement a single DF file for a single complex dial or tape display, the use of multiple dials or tapes in the same composed image may lead to complexities in the assignment of layers and assignment of layers to windows. Table 2: DF Development for a Numeric Readout UA specified considerations for modification of the CDS configuration when new numeric readouts are configured in the CDS. If a single DF is used the variable items in the DF may be moved to the UA Configuration.

Table 2: DF Development for a Numeric Readout UA

Configuration Catagory	Notes
Locations	While this can also be performed using UA configuration, the DF File will include the locations of all widgets used in the numeric representation.
Needles	The DF file can contain the symbolology used in the needle representation as a collection of other widgets, as a symbol, or as an image. Whatever the choice, this can be placed in a container widget -- a rotation widget would be used for a dial.
Bugs	In the DF file, Bugs are treated the same as a needle. The location of the containing symbology within the container would be offset from the needle.
Limit Indicators	Limit indicators can be implemented as a type of bug if the limit is dynamic. They can be implemented direectly in the DF when the limit is static. In some cases, a static limit may be placed in the DF using dynamic operationg to limit the need to change the DF in future configurations.
Tick Marks	The size and locations fo tick marks may be best placed in the DF file. They rarely change.

Common Symbology Approach Using Configurable Core Symbology

Information from the TSS

The data from the transport that might feed information to a numeric readout is shown in Table 3: Transport Data for a Numeric Readout UA. This information is generally available from the "model" aspect of Hosted Capabilities without the need to add specific user interface functions.

Table 3: Transport Data for a Numeric Readout UA

Configuration Catagory	Notes
Primary Values	The value displayed in the readout and depicted by the primary needle is usually a simple connection to a TSS value. Inside of the configurable core symbology component this value is assigned a location based on the radial or liniar limits of the display.
Bugs	A value to represent a bug is mapped the same way as the principal needle, including using the same limits to calculate location.
Validity	Validity inputs may be represented by one or more boolean values. These can be mapped directly to inputs representing different styles that may be used in the capability.
Limit	In some cases a limit may be a variable value. This can occur if the limit changes during different flight modes or other states. These values can be consumed from the TSS the same manner as the primary values.

Numeric Inputs

The ability to input numeric data into a system can include a number of user interface conveniences depending on the capabilities' of the platform. Dials for quickly entering frequencies or changing values have long been a desired input mechanism that might be available on some platforms. The use of multiple dials or nested dials for data entry is not something that a new capability should include in its user interface design. These entry mechanisms are tied to the Edit Box Numeric widget in the CDS, but there may be more complicated mechanisms available to a custom UA.

Numeric Entry may also include a commanded v/s actual display style. In these cases the actual value used in the system is provided as an input and the commanded value comes from the generic UA. When the values don't match, the display of the value could render the pending state of the commanded entry.

Configuration of the UA

The configuration parameters to consider as part of the numeric entry include the values specified in Table 4: Configuration for Numeric Entry in a UA. Of primary concern is the ability to configure fields in these entries. Entry of something like a Latitude or Longitude value can be broken into several fields. The use of an EditBoxNumeric widget may also be used.

Table 4: Configuration for Numeric Entry in a UA

Configuration Catagory	Parameters
Numeric Entry	Default Value, Format
Commanded Style	

Modification of the DF

Table 5: DF Development for a Numeric Entry UA specifies considerations for modification of the CDS configuration for numeric entry.

Table 5: DF Development for a Numeric Entry UA

Configuration Category	Notes
Edit Box Widget Type	The inputs from dials and keyboards are generally built into the Edit Box types within the CDS logic. Some, more complicated means of entry may require interaction with the UA through the transport.

Common Symbology Approach Using Configurable Core Symbology

Information to the TSS

Table 6: Transport Data for a Numeric Entry has suggestions for distribution of the resulting numbers.

Table 6: Transport Data for a Numeric Entry

Configuration Catagory	Notes
Entered Value	The entered value can be distributed using the same techniques as an input.
Validity Check	An input from the TSS could provide feedback if the entered value is acceptable.
Actual Value	An input on the TSS could provide a current value for display in commanded v/s actual format.

Text Display and Input

Alpha-Numeric Text fields distributed in the TSS tend to be focused more on the user interface than on the capability logic. Text beyond numeric readouts in a user interface tend to be static (Labels, Units) or come from an enumeration of possible known values (states, commands). Static text can be simply added to a DF of configured in a UA configuration file. Enumeration values would more likely be handled by a List Selection (see below). True Alpha-Numeric values include things like flight plan waypoint names, airport/VOR codes, and other identifiers.

TSS distribution of these values tend to include lists or lookups that are not as simple as the numeric examples above. There may be a Selected or an Active waypoint name value that would be easy to map; but there is an implied user interface part of the capability that supports these kinds of values. The ability to display text values and allow user input of those values may require more user interface development in the new capability, but this development is still avoiding the user interface complexity and promotes common symbology.

List Selection

The ability to select an item from a list is another common user interface feature that can be built around the integration style of the platform and be removed from the specific user interface design of a new capability.

List Styles

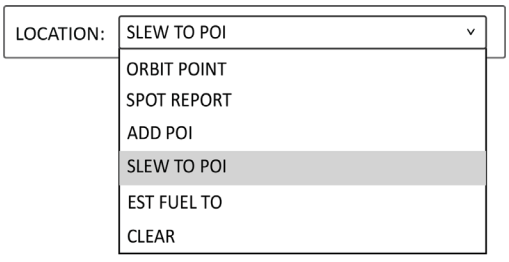
Lists can come in many forms. A common form is a simple list showing a selected item. The Combo Box presents a way to use such a list in a space saving means. The selection of a mode may also be presented as a list of buttons with only one of them selected (as a radio button list). More complicated tables feature columns and rows of data, such as a flight plan list that shows the distances and times related to upcoming waypoints.

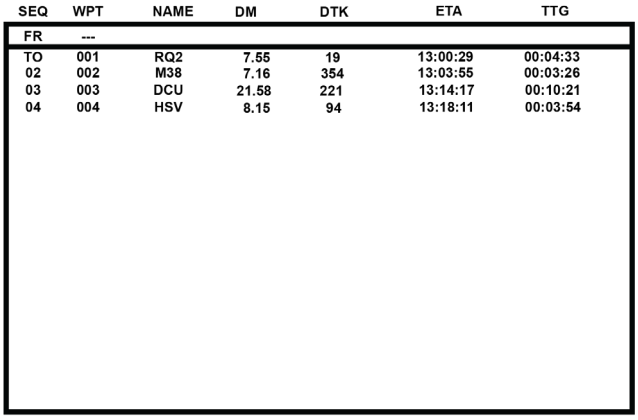
Item Selection in a list may not be included in some interfaces, while in others a single selection or multiple selection may be included.

Item Availability may be on or off. Some interfaces may remove items from the list, other may grey out unavailable options. The inputs for determining this state would be similar to the validity inputs for individual numbers.

Icons may be desired to represent different possibilities for some list columns. Icons would be used like an enumeration where an icon is configured for each enumerated value in the UA configuration.

Sorting of list data may be static per the configuration, or may be part of the user interface.





Enumerations

In some cases, the list selection is simply an indication of the current value of an enumeration. This can be constructed using a similar pattern to numeric entry on the TSS, with a configuration file representing the text associated with each value in the enumeration.

Common Symbology Approach Using Configurable Core Symbology

Table Values

When the displayed values take on the form of the more complicated text fields, or include the use of multiple related values, the mapping of transport data to the user interface becomes even more integrated into the capability. Here the TSS interaction is not as simple as the transforms of numeric and Boolean values used in our initial examples. List selection can also become quite complicated when considering multiple columns of data points. Yet, the addition of this common interface component can greatly aid in the control of new capabilities through these common components.

List Select TSS Considerations

The transport interfaces to the Configurable Core UA can be built to receive column data in a set of connections containing two values in an ID/Value pair. The connection would identify the column the data represents. The ID would relate the values in a row and the Value would provide the cell for that row/column. Use of separate connections would allow a generic input that can be a numeric, text, or Boolean data.

Alternatively, a single input can be used with ID, Column, and Value information. This input style may present some additional parsing of data within the UA if columns are to be displayed using limit or state information.

Example: A flight plan list includes the Estimated Fuel to each waypoint. Waypoints with the remaining fuel below a fuel reserve could display the fuel remaining value in amber. Use of individual ID/Value inputs can have one input set to RemainingFuel for Waypoint. A spate limit input to the UA may be connected to "Fuel Reserve". The logic in the UA is fairly simple. If the single input is used, the Value would likely have to be converted from text into a number for the calculation.

Approach to ARINC-661

Extending abstraction to graphic representation

Looking to extend the abstraction of graphic representation is one of the main points of this paper. By recognizing that there are some common concepts of presenting data to the user. The idea is to create a few sets of generic highly configurable applications to allow the building up of sets of these applications to work in concert and configured to present the data in the specific look and feel of the target system.

Separate the system requirements from the system design

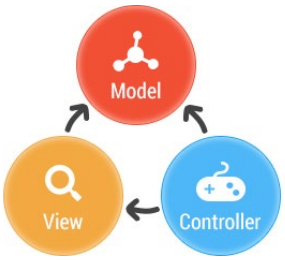
By using an ARINC-661 (A661) CDS we are already talking about separating the look and feel requirements of the system to the definition files used to configure the CDS. The UA is then responsible for the functional requirements, and will then modify the states or parameters of the widgets based on functional states of the system. Using the model, view, controller pattern, we should be able to use the Configurable Core UAs along with the CDS to provide most of the view and controller aspects of the capability. Ideally, new capabilities are just the model aspects with the view and controller supplied through configuration. There is more discussion of Model View Controller architecture in Architecture Approaches in Evolution paper (Edwards, Price 2020).

A capability can be added by:

a) modifying A661 definition files,

b) modifying the configuration of one or more existing core configurable UAs, or

c) adding another instance(s) of these core configurable UAs.



The TSS is then configured for any new connections. The more we use common core configurable UAs the easier it becomes to modify what can be displayed.

Configurable Core UAs and the CDS

ARINC-661 interfaces

The interfaces with a CDS are defined in the ARINC-661 standard, and used for communication in both directions between the CDS and the UAs. Within the definition files for the CDS are sets widgets which are the basic graphic primitives that can be displayed along with the parameters that can be changed for each widget, allowing for the updating of the images being displayed. There are layers which are collections of related widgets. Note that a layer may only be controlled by a single UA, but that a UA, may interact with multiple layers. Also, there is a controlling, or super layer that pulls everything together for the display(s). It is worth noting that these ARINC-661 messages will be passed back and forth through TSS connections between the CDS and the various UAs. The CDS also communicates with the GPU using graphics drivers.

Common Symbology Approach Using Configurable Core Symbology

CDS

The configuration of the CDS, to group sets of these core common configurable UAs together to act in concert as part of a larger overall capability is key. The versatility and the look and feel requirements are supported through the configuration of the CDS; including placement and layering of the correct widget types. The CDS itself should need no code modification.

Controller

A Controller is required to make this all work. The CDS itself controls what is displayed based on its state, and the messages it receives. As mentioned a bit earlier, there is a master or super layer that controls which set of layers are active and visible. There is a controller UA that is in control of the master or super layer. The choices made on that layer will govern what can be displayed at any given time.

Configurable Core UAs

A small distinct set of configurable core UAs would receive generic data and be configurable to allow that same data to be presented in a variety of ways, based on configuration. There could also be some generic interactions with the configurable core UAs.

A good practice would be to create a class for each set of generic interactions, then an overall generic display class (Tape/Dial/List/etc), so applications need not worry about the "type" of generic display. This abstracts even the more generic type into a truly generic display. Additions to the display can be made as required without impacting existing code

Configurable TSS

The ability to easily configure a TSS to support different messages is generally key to success and growth. With the idea that these configurable core UAs may support differing amounts of data based on configuration should only mean that more message of a specified types are being passed back and forth. It would require more connections to be supported by a TSS based solely on configuration of that TSS.

This is where message translation comes into play. The Hosted Capabilities have specific data that is defined in the user supplied data model and references the based on type of data, units, and has the semantics to know exactly what the data is used for. In contrast, the Configurable Core UA inputs are modeled as things like "Displayed Number" in order to be generic, the semantics are not important. These models measurement systems and ranges are based on the readout, including just enough information to present the generic data. The transformation of the Hosted Capability data to the Configurable UA generic should be handled as part of the transforms in the TSS configuration (Edwards, Price, & Tanner, Transformation Capabilities in Configurable Common Services, 2018).

Use Case Analysis

The UH-60M Crew Mission Station project developed a number of UAs for specific capabilities. This section examines how many of the CMS pages would have been impacted if the Configurable Core UAs were implemented.

EICAS Page

The EICAS Page for the UH-60 is generally a collection of numeric readouts as tape displays. Additional features include the Caution and Advisory Readouts, Engine Out Indicators, and some frames that change color based on the displays within the frames.

Readouts: The tapes and readout boxed on the EICAS page show limit ranges within linear tapes. The basics for the tape style readout would be used for all of the readouts and tapes on this page.

Caution and Advisory Display: The caution and advisory display amounts to seven different lists of enumerations tied to a text representation of the enumeration. An Acknowledged state indicates if the item should be shown with the background highlight. Had the system been built with a list mechanism, the state and id for each of the lists could be constructed either in a TSS transform, or in a light weight CAS PCS that simply managed the individual lists.

Indicators: The indicators on the EICAS page can be constructed as a set of widgets in containers that can be turned on/off. The frames that change color based on the values within would present an interesting configuration setting. If the values for a "displayed readout state" or a "displayed color" were output from the Configurable UA, a TSS transform could be written to combine these outputs into a set of frame states as inputs.

Had the EICAS page been developed using common core capabilities from the Rapid Integration Framework (RIF) software base would have been easily transferred to different aircraft as engines and fuel tanks change.

Flight Plan Page

The Flight Plan Page is simply a list of flight plan waypoint data. Some of the information presented on this page is not currently distributed in the system. Had this page been developed using a configurable core UA; the information on the page would be distributed, adding to the data available of other capabilities.

Fuel Estimation

The Fuel Estimation Page represents the most complicated UA that might have been developed using configurable core components. This page includes a few edit boxes for values that feed into an estimation function, a list of waypoints to pick from, and a results window. This UA was developed with the estimation calculations and the ARINC-661 data combined into the same application. If a system of configurable UA Readouts had been available the reusability of the resulting code would have increased. The calculations for estimation would have been separated into a separate function. This would have forced the Model/View/Controller pattern and led to more flexibility in the eventual system.

Conclusion

Use of Configurable Core UA sets for common user interface patterns can greatly reduce efforts to bring new capabilities to the warfighter. These core capabilities will also enhance the commonality in the user interface, allowing tighter integration with the platform.

Patterns for transforming TSS messages to these common core components as a capability in the TSS implementation drives additional need for the selection of a TSS.

Implementation of numeric readout capabilities is pretty straight forward. Lists that support display and selection of enumerated values are also a simple capability that can provide great benefits to common interfaces. Implementation of text display/entry and table views may require more complicated TSS interactions, but developing all of these capabilities would greatly reduce the need to develop specific user interface code for most new capabilities.

Use of these common capabilities can also support user interfaces that combine data from multiple sources. Bugs displayed on some numeric readouts may come from multiple sources. Developing these readouts following a configurable pattern can assist in extending these readouts to include limits, or related values without redevelopment (only a reconfiguring) of display capabilities. An approach to separation of configuration criticality could also allow high criticality numeric readouts to have low criticality indicators without impacting high criticality artifacts.

References

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

- AEEC - Engineering Standards for Aircraft Systems. (2019). *ARINC 661.*
- Edwards, C. J., Price, S. P., & Mooradian, D. H. (October, 2017). The Impact of the FACE Technical Standard on Achieving the Crew Mission Station (CMS) Objectives. The Open Group.
- Edwards, C. J., Price, S. P., & Tanner, W. G. (2018). *Transformation Capabilities in Configurable Common Services.* The Open Group.
- FACE Consortium. (24 Jun 2014). *Technical Standard for Future Airborne Capability Environment (FACE), Edition 2.1.* The Open Group. Retrieved from www.opengroup.org/library/c145
- FACE Consortium. (December 2017). *Technical Standard for Future Airborne Capability Environment (FACE), Edition 3.0.* The Open Group. Retrieved from www.opengroup.org/library/c17c
- PEO Aviation. (2018). Rapid Integration Framework (RIF) Demonstration Information Packet. *Proceedings of the 2018 September US Army FACE™ Technical Interchange Meeting.* Huntsville, AL: The Open Group.
- Price, S. P., & Edwards, C. J. (2017). A Common Command Interface for Interactive FACE Units of Conformance. The Open Group.
- RTCA, Inc. (2012). DO-178C, Software Considerations in Airborne Systems and Equipment Certification.

About the Author(s)

Christopher J. Edwards has been working in the avionics industry for over 25 years, primarily on cockpit systems for military aircraft. In those years, he has served in leadership roles in System Architecture, Software Development, Requirements Capture, PVI development, Qualification Testing, and Project Management. Mr. Edwards work within the FACE Consortium has been as a principal author on both the FACE Conformance Policy and the FACE Technical Standard as well as many other consortium documents. Mr. Edwards currently leads the FACE Conformance Overview presentations and serves as a co-lead of the FACE Technical Working Group (TWG) Conformance Verification Subcommittee and as the facilitator of the FACE Verification Authority Community of Practice. Mr. Edwards serves as a MOSA Subject Matter Expert and is the Chief Architect and Systems Engineer for the Fixed Wing Family of Systems as well as other RIF related projects.

Steven P. Price has been working in avionics and embedded software for more than 30 years. He has worked on several different graphic user interfaces, including cockpit systems. He has been a leader in the design and implementation of some of these systems, along with being involved with the testing of some of these systems. Currently, Mr. Price is one of the Principal Software Engineers for the Rapid Integration Framework (RIF) and the principal developer of the Crew Mission Station (CMS) Menu System. He is a co-lead on the FACE Transport sub-committee and FACE Verification Authority Subject Matter Expert (SME), along with involvement in other FACE sub-committees.

About The Open Group FACE™ Consortium

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

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

About The Open Group

The Open Group is a global consortium that enables the achievement of business objectives through technology standards. Our diverse membership of more than 800 organizations includes customers, systems and solutions suppliers, tools vendors, integrators, academics, and consultants across multiple industries.

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

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

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