A method for defining rule elements based on civil aircraft interface development process

Jiejing Zhang¹, Xinyu Zhang²*, Xinai Zhang³, Dake Guo⁴, Haomin Li⁵,
¹Commercial Aircraft Corporation of China Ltd, Shanghai, China
²Commercial Aircraft Corporation of China Ltd, Shanghai, China
³Commercial Aircraft Corporation of China Ltd, Shanghai, China
⁴Commercial Aircraft Corporation of China Ltd, Shanghai, China
⁵Commercial Aircraft Corporation of China Ltd, Shanghai, China
*Corresponding author’s e-mail: zhangjiejing@comac.cc

Abstract: In the process of civil aircraft design and development, interface definition is a very important and complex process element. In this paper, a rule element definition method based on civil aircraft interface development process is proposed, and the rule element definition to support the interface development process is given. Rule elements provide useful input on structure and content for civil aircraft interface definition activities, especially for low-level interface definition activities. All levels of an interface definition can be related to rule elements, but lower-level interface definitions have the greatest amount of data, and most of the definitions are done in lower-level interfaces. For the design activities of the stakeholders in the low-level interface development process, these rule elements are equivalent to the formulation of multidimensional and unified standards and requirements, so that the stakeholders of interface definition (main manufacturer and suppliers at all levels) can carry out coordinated interface development activities under such standards and requirements.

1. Introduction
The development of modern civil aircraft is an increasingly complex and highly integrated process, and the integration of multi-complex systems has become one of the characteristics of the development of civil aircraft. In the design process, the main between manufacturers and suppliers, vendors and suppliers, the need for interface design and development work to carry out a lot of discussion and coordination, which requires a set of perfect process specification to be reasonable and effective control, which requires a set of perfect process specification for reasonable and effective control. In the process of controlling the interface design process, it is very important to define the elements of interface development rules.

In ISO-IEC 1220-2005 "Systems Engineering -Application and Management of the Systems Engineering Process", the related terms of interface development are mainly defined, such as defining interface, defining functional interface, defining physical interface, interface management and so on. In ARP 4754A (Guidelines for Development of Civil Aircraft and Systems), the formation of functional interfaces and the output of interface files are described and explained. In NASA's Systems Engineering Handbook, interface files are defined.
In order to more effectively follow the requirements of the above standards in the implementation process of civil aircraft project, this paper proposes a method of defining rule elements based on the development process of civil aircraft interface.

2. The purpose of defining the rule elements of the interface development process

Reasons for using rule elements include:

a) Specify checks and develop tools
b) Early validation and control
c) Reduce the load for inconsistency analysis
d) Harmonize
e) Interface definition, communication medium and way to use it
f) Common guidelines
g) Support interface definition for all actors
h) Interface definition uniqueness

The details are shown in the following figure:

![Figure 1 - Main rationales for defining rules](image)

3. The type of rule element

Rules are aircraft program dependent. This section provides indication and guidelines about rules types that have to be considered. Rules generally correspond either to:

a) An a priori knowledge on interface objects that could be used to make control and checking on these data.
b) Constraint introduce for harmonization purpose and simplifies the interface design.

At low interface definition level each communication item is identified and defined.

a) It is identified by its name
b) It is defined by its features and attributes.

Typical example of this is:
Virtual Link. VL name and its definition shall be unique.

Signal name and definition

Another key objective is that the low level interface definition is be managed by tools. For specifying tools, assessment and rules are necessary.

Typical example of this is Maximum size of a string for a name (see hereafter).

Two main types of rules could be considered:

a) Rules dealing with a single field
b) Rules dealing with a several fields

The first one, for instance string maximum length, is the most simple to fit to and check. The second one is based on cross dependencies between several fields and often requires a tool to be checked (e.g: checking the relation between the Virtual Link Identifier and the value of the MAC destination address).

3.1 Interface definition file/document name

The first rule is a naming rule on the name of the interface definition itself. There are several interface definition files. At least include:

a) FICD (Functional Interface Control Document)
b) EICD (Electroincal Interface Control Document)

It was suggested that despite several levels of definition (functional, logical) are included in FICD a single FICD document is used per package, distinguishing the level of details thanks relevant sections.

For organization purpose a naming rule is recommended for all these files. As each of these files represent either a system or a subsystem/package interface definition point of view, the name of the file could be prefixed by the name of the system/package under study.

3.2 Mandatory fields

Low level interface definition may have the shape of a set of tables with different sheets with specific fields.

One of the most basic rules to fulfil these fields is to specify whether a field content is mandatory or optional. For instance the address of a signal in a message or label is mandatory while the same fields are used as comment fields to have some “nice to have” helping the interface object understanding.

It is a significant support for the system designer to identify which field are mandatory and which one are nice to have, when populating the interface definition.

3.3 Naming rules

Low level interface definition deals with data communication items and objects. These objects are generally identified by their name. So, rules on name are important and there is a need for harmonization and avoiding mismatch when talking about a given communication object.

Once more the purpose is to enforce consistency as soon as possible. Checks are foreseen and consistency issues will be identified, but in a development process (especially when suppliers are in the loop) the early detection of inconsistency issues is very important in term of maturity, delay and cost. In this frame, naming rules are of relevant help.

3.3.1 Syntax

Names (for interface item identification and/or definition) have the shape of a string, made of a set of characters taken from an alphabet. This alphabet has to be defined. It is the set of allowed characters.

a) Authorized characters: As an illustration, let’s take an example: Are blank characters, <space>, star “*”, question or exclamation marks “?”、“!” or other special characters authorized?
b) Enforcing the set of authorized character for the first character of the name (e.g: a letter): The answer is often driven by the constraints about tools that will manage the interface definition data. Restricting the set of allowed characters is a efficient way to enforce consistency, avoid erroneous naming and simplifies the checking of interface items names.
c) Maximum string length: It is the maximum number of characters in the string for naming the a
interface item (including attributes).

### 3.3.2 Harmonized prefix

It is a relevant interface definition rule that naming rules for communication items take benefits of this principle and identify the name of the publisher in the interface item in the first part of the name (prefix). For instance, it may be several letters identifying the application name publishing the interface item under definition.

This kind of rule simplifies the checking by the system or package designer responsible that the signal is unique for its system or package.

This principle may prevail for communication media for which only a single transmitter is allowed as for instance:

- a) ARINC 429 Buses
- b) ARINC 664/ Virtual Link names
- c) ANALOGUE / DISCRETE Lines
- d) ARINC 825 line names

### 3.4 Name uni-city

One of the main topics about in term of consistency is uni-city. For instance, two different buses shall have different identifiers and names. Two different Reception ports on a subscribing application side shall have different identifiers and names in order to store and distinguish the different received values. Different signals embedded in a message of Label shall have no addresses (location) overlapping.

This rule shall be respected during the interface definition phase. These addressing rules will be verified by tools in a further step.

### 3.5 Reference List

It is a very useful practice to support system designers (and supplier package designer as well) to define naming rules and reference lists.

Reference list is a set of allowed names for a given aircraft program (common names space).

It allows enforcing consistency (harmonization, uni-city, fitting subscribed interface items name with published one, as soon as possible and avoid names mismatch thanks a predefined list.

This list may increase over the time according aircraft program maturity but to do so a particular process shall be respected by delegation. Hence it allows an early validation and agreement at high level of the updated list, for slow changing interface items.

Reference List is a kind of small database applicable for names, interface items which are candidates for Reference List are:

- a) Application names
- b) Systems names
- c) Equipment or hosting resources names
- d) Connector and PIN names

Reference List is a possible implementation to restrict the field of possible name space.

A generalization of reference list is to achieve an highly structured definition of the associating System, Application, Equipment, names and number of occurrences.

This kind of aircraft level architecture summary for communication interface, could be initialized at the beginning of the aircraft program and be updated all along the architecture and design become more mature. A recommendation is that changing this reference list should require to follow a relevant process (with associated authorization).

### 3.6 Enumerate fields

When possible attributes’ values of an interface belong to a finite set of values, it is useful to define the type of the attribute as enumerate. Several reasons may justify that only a finite set of values are allowed, for instance constraints due to a standard.
A simple example is that of ARINC 4299 buses, which may be low or high speed. No other values are foreseen. An enumerate Low/ high is relevant in this case for the bus speed attribute.

Another example is the protocol attribute used for the data communication in case of protocol communications. Generally, a finite set of appreciative protocols are defined (ARINC 615-A for data loading, specific protocols for maintenance, etc…). Once more this use case could be supported by a relevant enumerate type for the protocol name.

Of course, in the case of enumerate type the rule is that the final attribute value set belongs to the associated enumerate set.

3.7 Attribute range

Values of attributes are not infinite and a range has to be specified and enforced for numerical attributes (e.g: the value of a port identifier). The associated rule specifies the authorized range.

3.8 Units fields

It is very useful to harmonize and constraint the units of fields. For instance for a “transmission rate”, enforcing the unit as “millisecond” rather than letting it open (second, ms, µs) will significantly improve the maturity and consistency of the interface definition. The rule in this case it to provide the value in relation to the foreseen/ enforced unit.

3.9 Values consistent with standards or mutually dependent

Some values of interface definition items are constraint by the standard itself.

Generally, standards bring constraints for instance for structured data. This is for instance the case of ARINC 429 label, which are structured according fields (label number, SSM, SDI, parity bits). This foreseen structure brings rules to be specified, followed and checked.

An example is the MAC source/ destination addresses or IP source / destination addresses. All of them have a format and part of content enforced by the ARINC 664-part 7 standard itself.

In the interface definition files, field associated to these addresses should be in line with the standard policy as illustrated on an example for IP unicast addresses in A664 p7 standard (section 3.4.1.4.1) reminded hereafter.

![Figure 1 – Example of an addressing constraint in A664 p7](image)

The definition of IP addresses in use in A664 p7 protocol is strictly policed and could be easily specified as a check for a tool. The System designer has to respect this compliance to standard when defining the interface of its system.

Some dependencies between MAC, IP addresses and VL identifiers exist. These dependencies between different field in the interface definition could be considered as rules. They should be considered when designing the interface and early verified to achieve interface definition maturity.

Another example of constraints coming from the standard is the A664 p 7 message format, as far as A664 messages are highly structured data. They shall fit for instance alignment constraints and message padding (e.g: cf A664 part 7: 1.4.1 and 1.4.2 Message Format Example).

These rules are well specified in the A664 p7 standard and could be easily formalized and implemented in tools to supports system designer in defining the messages structure, which is a
Another type of rule deals with the order used for fulfilling low level interface definition. The point is to define the higher-level definition first and the lower level of definition in a second step when there is a dependency between the fields.

For instance, a message or label structure shall be defined and fulfilled in the interface definition only once the communication media (e.g: ARINC 429 and ARINC 664 p7) has been chosen and defined.

In the same way of upper level information defined first:

a) The system publishing a parameter shall be defined before the equipment.

b) The Equipment publishing a parameter shall be defined before the application.

Another well know rule for fulfilling fields in interface definition is: Publishing fields content shall be defined and fulfilled before the associated subscriber’s fields (is the other system/ package / equipment interface definition) content identification.

3.11 Consistency between interface definition part of several concurrent process

It has been highlighted that some interface information are used by different processes: A typical example is the line name used in the interface definition process and also in the wiring definition process. Line name is common to both process and the fact that its value is common and not erroneously duplicated shall be verified as soon as possible, either by rule, process of both.

4. Conclusion

In this paper, a definition method based on rule elements in the development process of civil aircraft interface is proposed, which restricts the coordination and unified management in the definition process of civil aircraft interface, effectively promotes the reliability and effectiveness of the development of civil aircraft interface, and improves the efficiency.

5. Thanks

Thank you to the members of COMAC’s requirements and interface management team for their inspiring work. At the same time, thanks to the judges of this paper, their opinions to better promote the quality of this paper.

References

[1] ISO-IEC 1220-2005, Systems engineering-Application and management of the systems engineering process.
[2] SAE ARP4754A, Guidelines for Development of Civil Aircraft and System.
[3] DOT-FAA-AR-08-34, Requirements Engineering Management Findings Report.
[4] FAA, System Engineering Manual.
[5] EIA-632, Processes for Engineering a System.
[6] INCOSE, System Engineering Handbook.
[7] NASA, System Engineering Handbook.
[8] ARINC 664 PART 7, Aircraft Data Network Part 7 Avionics Full Duplex Switched Ethernet (AFDX) Network
[9] ARINC 429, Digital Information Transport System, DITS
[10] ARINC 825, The General Standardization of CAN for Airborne Use