"EXTENDING THE SCALE OF OUR MODELLING ENVIRONMENTS WITH CFS AND SEDS"

Zahi Manzelji [(1)], Franco Bergomi [(1)], Régis de Ferluc [(1)]
(1) Thales Alenia Space, France, E-mail: zahi.manzelji@thalesaleniaspace.com, franco.bergomi@thalesaleniaspace.com, regis.deferluc@thalesaleniaspace.com

Abstract: This paper presents the work that was led by Thales Alenia Space to take cFS (*"core Flight Software"*) and SEDS (*"SOIS compliant Electronic Data Sheets"*) standards into account in its model-based On-Board Software (OBSW) design environment, CCM4Space, in the frame of Lunar Gateway programs.

Keywords: PUS, cFS, SEDS.

1. INTRODUCTION

Up to now, European Space Missions has been largely based on the ECSS PUS standard (*"Packet Utilization Standard"*). In the frame of Lunar Gateway programs, Thales Alenia Space France is involved in the development and manufacturing of HLCS (*"Halo Lunar Communication System"*), I-HAB (*"International Habitation Module"*), and ERM (*"Esprit Refuelment Module"*), in a worldwide collaboration initiated by the NASA (*"National Aeronautics and Space and Administration"*). This context has brought up the need of an harmonization of the on-board software architecture. NASA's cFS has been chosen as common framework, and is imposed to every manufacturer and contractor working on the Lunar Gateway development. While PUS and cFS share many common aspects, they impose two radically different paradigms in terms of software architecture. In order to be able to work efficiently – and to meet up the strong planning requirements on the three programs, Thales Alenia Space France updated its model-based OBSW design environment to make it compliant to both PUS and cFS standards. A first part of this document presents the evolutions that were made to the design environment to handle cFS.

Working worldwide on Gateway modules that will have to interact and communicate between each other also imposes precise interface definition and specification. SEDS interface definition standard has been chosen on all Gateway programs to document software components (i.e. cFS applications) interfaces. A second part of this document explains how SEDS takes part in the software development lifecycle, and what tools have been developed to interact with this new standard.

2. CCM4SPACE

2.1. PRESENTATION

Thales Alenia Space uses CCM4Space for all On-Board Software design activities; CCM4Space is a customization of a Thales Group proprietary tool called Melody CCM which provides editors to define architectures for Software Systems based on Component-Based Software Engineering concepts.

Its core meta-model,CCM (Corba Component Model) is extended with domain-specific extensions; For instance Thales Alenia Space France has developed extensions in order to cope with the domain-specific concerns such a PUS, Precise Data and Interfaces design, Ground-Board communication, etc…

Melody CCM is also augmented with code generators (Ada , C) and other transformations (TM/TC IDS (Telemetry/Telecommand Interface Data Sheet) for instance), altogether composing the Software Factory (CCM4Space) used by software engineers to optimise on-board software development. This tooling is mostly aligned on the On-board Software Reference Architecture (OSRA) of ESA.

2.2. PUS STANDARD COMPLIANCE

The ECSS PUS standard defines the composition of packets transiting on the ground/board link. Many services are addressed such as command acceptance, housekeeping, event reporting, memory management, FDIR management, …

Traditionally, models and model elements in CCM4Space are used so-as to reflect very closely the PUS standard..

3. CFS FRAMEWORK

3.1. PRESENTATION

NASA's on-board software architecture is based on the cFS (core Flight Software) framework. This one relies on several operational concepts :

- Kernel services (CFE – Core Flight Executive)
- Independent and reusable applications
- A software bus routing messages (CCSDS packets) between applications

Therefore, a set of common cFS applications is provided to manage the some monitoring and control functionalities than the PUS standard services (see 2.2).

The main difference between a cFS architecture and our legacy on-board software architecture is the way applications communicate between each other.

While our design traditionally relies on synchronous or asynchronous calls of services specified in interfaces provided or required by application components, in a cFS architecture applications are communicating using publish/subscribe communication patterns, with all communication going through the software bus.

Instead of direct connections between application components, the software bus intermediate sever makes all applications independent from one another.

To be noted that cFS also make possible to have libraries (cFS libs) which can be accessed in a direct way by cFS apps.

3.2. MODELLING CFS APPLICATIONS

To describe the typical publish/subscribe interactions between cFS applications, and to keep the possibility to have direct interaction patterns (when for instance an architecture contains cFS libs which are accessed in a direct way, i.e. using interfaces, facets and receptacles), new concepts have been introduced in CCM4Space, using CCM standard "generic interactions" capabilities.

Practically, this part of the standard allows to define your own interaction patterns, using connectors with ports and exchanged data.

In our case, to model the publish/subscribe interactions between cFS apps and the software bus, we defined a generic PublishSubscribeConnector, with two ports (subscription/publication) and one exchanged data corresponding to the published/subscribed packet payload structure. This generic connector is then instantiated for each packet type exchanged on the software bus.

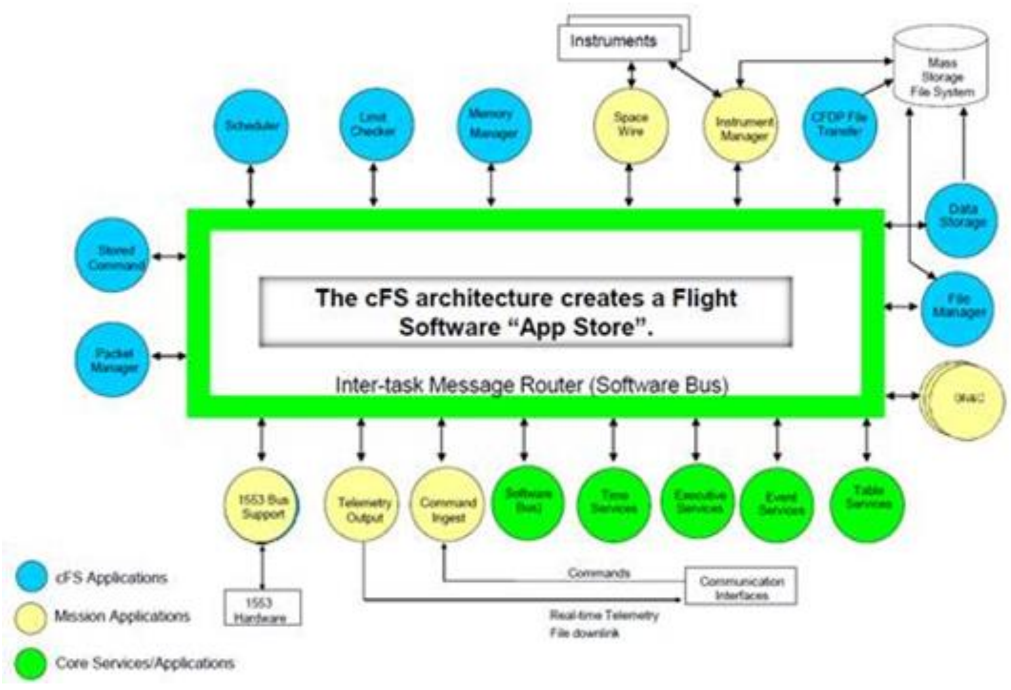

Figure 1: Example of cFS architecture

Each cFS app interacts the same various ways with the software bus :

- Receiving telecommands (packets subscription)
- Sending housekeeping (packets publication)
- Emitting events (packets publication)
- Being configurable via adjustable data (using cFS tables which involves specific telecommands)

As data is always exchanged via packets with the same header but a different payload, exchanges can be modelled in a standardized way, and this modelling can even be automatized. The only modelling effort that needs to be done concerns the structure of the here-above listed elements.

This is performed using classical CCM structure modelling, in a first model that we call *packets*.

A generator then updates all communication layer model elements in separate models.
For each packet defined, a connector is instantiated with the corresponding payload structure as packetType, in a second model that we call *interface*.

Thirdly, for each packet defined, a port with the correct direction (subscription for telecommands, publication for other packets) is added to the component corresponding to the cFS application. This port references the previously instantiated connector type. This is stored in a third model that we call *component*.

Finally, as in common architectures modelling, components are instantiated and deployed on a target. The major difference here is that instances of connector types are also instantiated and deployed on a target. Components and connector instances ports are connected. This way we guarantee that applications do not see whom they are communicating with.

4. SEDS

4.1. PRESENTATION

Spacecraft Onboard Interfaces Services (SOIS) compliant Electronic Data Sheets (SEDS) describe data interfaces for flight hardware. SEDS describes these interfaces using machine-readable Extensible Markup Language (XML) to support the lifecycle of space vehicles.

On Gateway programs HLCS, I-HAB and ERM, SEDS standard is used to share cFS applications interfaces between the stakeholders. For instance, NASA delivers core cFS applications to all manufacturers, and delivers SEDS to describe those applications interfaces. At Thales Alenia Space France, to model our own cFS applications and describe their interaction with cFS ones, we need to have a model for each NASA application. SEDS are used to automatically initiate those models.

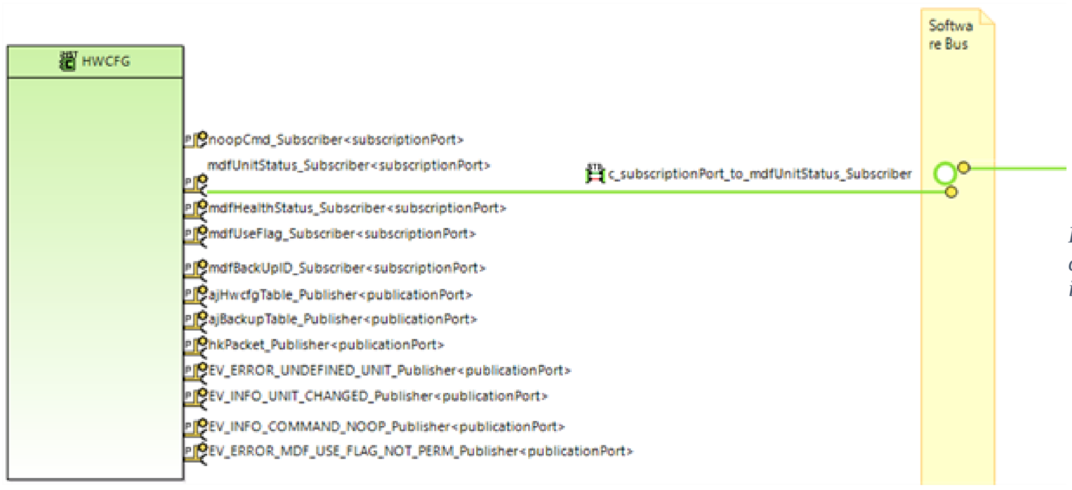
HWCFG
noopCmd_Subscriber<subscriptionPort>
mdfUnitStatus_Subscriber<subscriptionPort>
mdfHealthStatus_Subscriber<subscriptionPort>
mdfUseFlag_Subscriber<subscriptionPort>
mdfBackUpID_Subscriber<subscriptionPort>
ajHwcfgTable_Publisher<publicationPort>
ajBackupTable_Publisher<publicationPort>
hkPacket_Publisher<publicationPort>
EV_ERROR_UNDEFINED_UNIT_Publisher<publicationPort>
EV_INFO_UNIT_CHANGED_Publisher<publicationPort>
EV_INFO_COMMAND_NOOP_Publisher<publicationPort>
EV_ERROR_MDF_USE_FLAG_NOT_PERM_Publisher<publicationPort>
c_subscriptionPort_to_mdfUnitStatus_Subscriber
Software Bus

Figure 2 : Deployment of component and connector instances

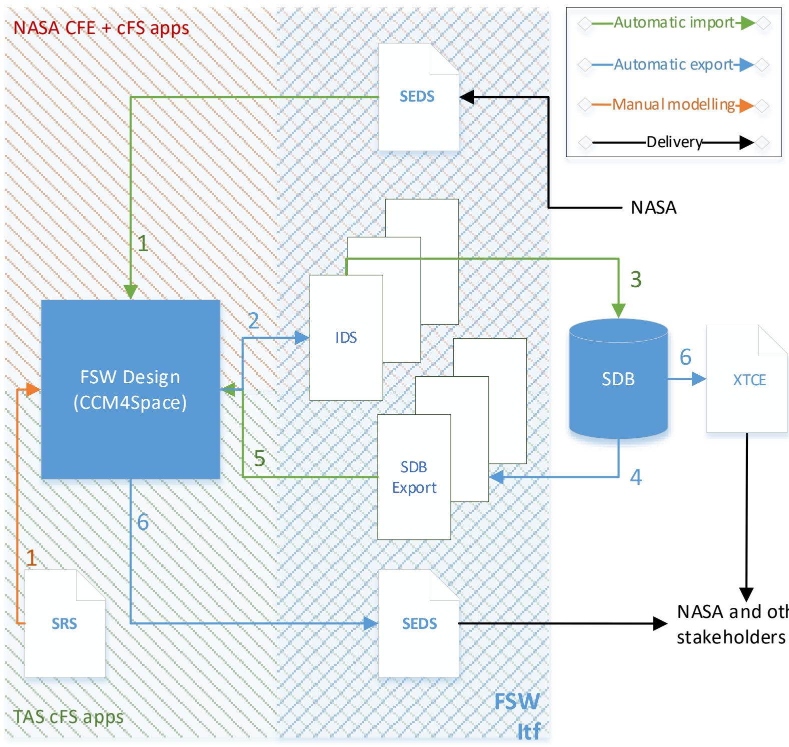

Figure 3:SEDS and OBSW (FSW) development process

SEDS TO CCM IMPORTER

cFS application SEDS files are used to automatically initiate CCM models. SEDS standard is quite permissive, and to that extend, each metaclass has a large number of attributes.

As a first step of the process, SEDS metamodel has been simplified to keep only what was considered useful to represent cFS applications.

This simplification consists in a second EEDS (Efficient EDS) metamodel.

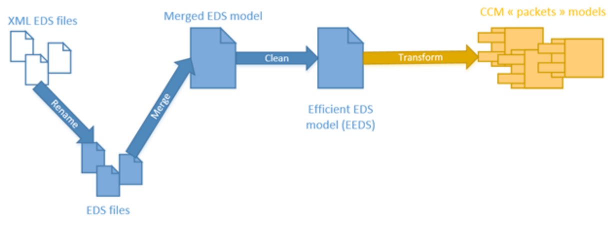

Figure 4: SEDS to CCM general import process

After some minor operations on input SEDS files to transform them into EMF EEDS models, the main transformation is performed by a Transposer (Transposer is a model-to-model technology part of the Polarsys Kitalpha Eclipse project) bridge between this EEDS models and CCM models representing cFS applications (*packets*, *interface*, *component* and *deployment*).

5. CONCLUSION

In the context of the three Lunar Gateway modules where Thales Alenia Space France is involved, adapting the Software Factory at minor cost was mandatory to ensure OBSW deliveries on schedule.

Thanks to the extensibility of our Model-Based OBSW Design Environment, which is permitted by the extensibility of CCM standard, we have successfully managed adapting our modelling practices and tooling to be compliant to both PUS and cFS standards.

Since those two approaches address approximatively the same needs in two different ways, we have been facing difficulties to change our habits in terms of architecture and development. However, we finally found advantages in cFS, like the standardization of exchanges between applications and the software bus, making it possible to minimize what needs to be modelled manually (i.e. exchanged data), and what can be automatically generated (framework-related elements).

The use of SEDS to represent software applications interfaces has also proved to be an efficient mean to share this information through automated (import/export) processes. Moreover, the expression capabilities offered by SEDS grammar, and the fact that its interaction with our modelling environments (Capella, CCM4Space) has already been secured during SAVOIR Electronic Datasheet Definition Use Cases [1], simplified a lot the development efforts on our tools.

6. REFERENCES

[1] "*[D14-A2] EDS Use Cases Implementation Report - TASF*", Thales Alenia Space