Technology Harmonization – Developing a Reference Architecture for the Ground Segment Software

N. Duro, F. Moreira, J. Rogado, J. Reis
Critical Software
Polo Tecnológico de Lisboa, lote 1
Estrada do Paco do Lumiar
1600-546 Lisboa
Portugal
nuno.duro@criticalsoftware.com

Nestor Peccia
ESA
Robert-Bosch-Str. 5
Darmstadt
Germany
nestor.peccia@esa.int

Abstract—[1][2]Over the past few years, efforts have been made in Europe to harmonize the space technology. Initially, the scope of the Ground Systems Harmonization roadmap was restricted to Mission Control and EGSE systems, but it was later broadened to include all Ground Systems Software.

The characteristics of the systems demanded the specification of the architecture to be based in a framework for distributed systems. The framework adopted, based in on the RM-ODP (Reference Model for Open Distributed Processing) standard and on the RASDS (Reference Architecture for Space Data Systems) initiative, was customized in order to follow the current Service Oriented Architecture (SOA) technological trend.

This framework has been applied to the Ground Systems software with the support of an UML modelling tool. In this process, four Viewpoints were used to describe the reference architecture.

The Enterprise Viewpoint, which captures the high-level system requirements, was based in the ECSS E-70.

The Information Viewpoint identifies information flows between main functional entities and describes the systems' Information Objects.

The Computational Viewpoint identifies services based on the decomposition of the systems into low granularity functionalities, accessed through well-defined interfaces. These interfaces represent invocation relationships between services, and are also characterized by dependencies to Information Objects defined in the Information viewpoint.

The Engineering Viewpoint represents the service distribution into specific components, which are executed across processing nodes (centres, computational resources). The interactions between nodes were modelled by the introduction of communication channels.

The Viewpoints definition introduces a formal methodology for describing the Ground Segment, which provides a better understanding of the systems functionalities and interactions as well as a consolidation of the terminology. The usage of a UML tool for the specification enforces the global model coherency, provides a user friendly navigation across the different viewpoints, allow the automatic generation of documents and creates traceability between objects. All these factors improve significantly the readability and comprehension of the architecture.

The model created will provide various advantages in new system developments. It will enable a better interoperability and completeness of the systems being designed and developed and also will greatly benefit the systems' maintainability namely the modularity, changeability and portability.

Future developments and enhancements to this architecture include namely the specification of interfaces of key components, which will enforce the compliance of new products with the reference architecture.

The architecture is bringing direct benefits to harmonization process, especially by allowing true interoperability between European space segment initiatives.

TABLE OF CONTENTS

1. INTRODUCTION TO HARMONIZATION

Ground Systems Software is one of the technical subjects of the European Technology Harmonization. Started in the second semester of 2002, it was initially focused on Mission

[1] 0-7803-8870-4/05/$20.00© 2005 IEEE
[2] IEEEAC paper #1134, Version 1.14, Updated December 15, 2004

Control Systems and Electrical Ground Support Equipments (EGSE). However, after the initial roadmap meeting, the scope was broadened to cover all Ground Systems Software including also Flight Dynamics, Mission planning, Simulators, data Archiving and Distribution.

One of the most promising paths for the harmonisation of Ground Systems Software is achieved through the identification of the components that present a high degree of commonality across different system implementations (or missions), and their progressive standardisation by clearly specifying their functionalities and interfaces.

According to these principles, a reference architecture has been defined, which allowed an exhaustive identification of the main Ground Segment (G/S) common services, spanning throughout different missions and also different mission phases.

This paper describes the innovative methodology that was used to design the above mentioned reference architecture, and presents the major benefits of such approach.

First it introduces the architecture framework applied and then describes the specification G/S Reference Architecture as well as the future steps to be followed. Finally, a few considerations are made regarding the benefits of the approach.

2. ARCHITECTURE FRAMEWORK

The degree of complexity of the systems that were analysed put important constraints on the methodology used for approaching the architecture specification. One of the first was to base the specification on a recognised and well adapted framework for modelling distributed systems.

Although a few architecture frameworks initiatives were available ([2], [3], [4], [6]) some did not present a satisfactory degree of completeness with regards to the set of objects used to model system abstractions, while others, more complete, were too abstract to effectively describe the intrinsic aspects of the systems analysed.

Therefore, it was found convenient to define a specific framework, more adapted to the kind of approach that was envisioned.

The framework, which is described in this section, is based, for its core principles and layered approach, on the RM-ODP (Reference Model for Open Distributed Processing) standard [2], to which a few extensions taken from the RASDS (Reference Architecture for Space Data Systems) initiative [3] were added, namely in the aspects related to object definition. On top of these basic principles, a specific set of abstractions was built, in order to provide the reusability and extensibility of components. This approach is inline with a Service Oriented Architecture (SOA) approach.

The first aspect of the framework is to introduce a set of specific Views, which are derived from the RM-ODP Viewpoints. In this approach, the following Viewpoints[3] are defined:

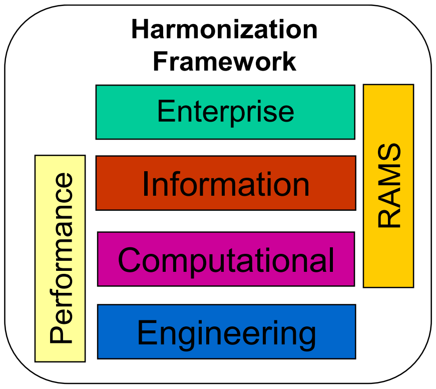

Figure 1: Ground Segment Views

Enterprise Viewpoint

It describes the Ground Segment specific objectives, entities, scope and boundaries from the organizational high-level point of view.

Information Viewpoint

The objective of the G/S Information Viewpoint is to describe the Ground Systems from the perspective of the information representation, and the mechanisms that govern the creation, reception and exchange of information between G/S functional entities or services derived from Enterprise Objects.

The information is described from the point of view of its structure and syntax, contents and semantics, and the relationships that exist among objects, the rules for its usage and transformation, and policies and constraints for accessing it.

In order to adapt RM-ODP to the Ground Segment specificities, a few additional concepts were introduced, namely for dealing with data representation diversity, data distribution needs and specific archiving functionalities.

Information Objects: the basic entity for describing the entities addressed in the Information Viewpoint is the Information Object, which encapsulates the concepts and is functionally associated with data representation, storage (archiving) and distribution.

[3] In this context, the term Viewpoint is used whenever the aspect described is identical to the RM-ODP one. View is used to describe an alternate aspect, which is particular to the G/S architecture description.

Information Objects are defined by a set of attributes that represent the constraints of the system, their relationship with other entities, their state and structure, and the pre and post-conditions governing their change of state. These attributes are:

- *Core capabilities*: Contents (data); Representation: structure and semantics; Preservation and Description Information; Data aggregation; Persistence and integrity; QoS Requirements (data storage and data transfer capacity); Access control; Event handling; Constraints

- *Ingestion Interface*: accept information (data stream, events, commands, parameters, etc…); publish; subscribe[4];

- *Distribution Interface*; Deliver information (data stream, events, commands, parameters); Request / Response; Distribute;

- *Management Interface*: Metadata handling; Flow and Access control; Persistence / Integrity checking; Reporting; Constraint enforcement.

The adopted representation of the Information Objects is based in UML as shown in the figure.

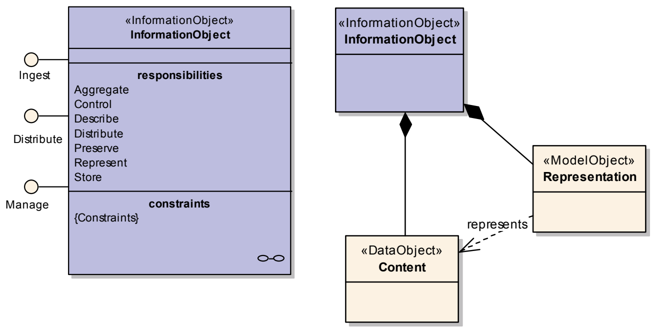

Figure 2 - Information Object Representation and Internal Structure

The Information Object internal structure, supported by the core capabilities, provides a separation between information content and representation. It can be explicitly used by instantiating either the DataObject, which relates to content (i.e.: data), or the ModelObject, which relates to the content structure and semantic representation (i.e.: metadata).

[4] From the point of view of the information object (which is by nature a passive object), Publish and Subscribe can be considered as included in the Ingest interface, since both correspond to insertion of information into the object container. Subscribing implies storing the subscriber identity in the object metadata. Publishing implies ingesting the new object content, which will be further distributed to subscribers by the associated Service.

Computational Viewpoint

The objective of the G/S Computational Viewpoint is to capture the lowest level of functional elements that can be represented in the Ground System application environment, independently of their aggregation in higher-level functional components. Otherwise, it is in essence an ODP Computational Viewpoint that captures the system functionality independently of its location and implementation aspects.

Service Concepts: leveraging the ODP standard, the concept of service has been introduced to support the computational viewpoint specification. The service is a software modelling abstraction widely used in several areas, namely in Software Engineering and Distributed Systems. A service can be defined as a basic functional element in the Ground System application environment that:

- Possesses an autonomous functional behaviour;

- Supports a producer consumer interaction model, delivering clearly identifiable results;

- Can be accessed through a well-defined interface, that supports location independence capabilities on request;

- Represents a potential degree of re-utilization across higher level elements;

- Makes sense from a business, representation and implementation point of view.

The properties enumerated above, allow the definition of a model from the perspective of G/S operations and processes, rather than applications and programs. This concept is the same as the one used in a Service Oriented Architecture.

The Computational Viewpoint describes the functional structure of the system in terms of services, by specifying their behaviour and their interactions. This is achieved by constructing a model that represents the service (the service object), and the logical connections existing between them (the interaction interfaces).

Service Objects: Objects are defined by a set of attributes that depict their behaviour. These attributes are:

- The *Responsibilities or requirements*, which govern the service implementation.

- The *Service Handle*, which uniquely identifies the service for publishing and binding purposes.

- The *Management interface*, which allows all operations related with service set up, registration and configuration of service specific properties.

- The *Access interface* (or Provided), which grants access to the service exported functionalities.

- The *External interface* (or Required), by which the service invokes other services in case of service composition.

- The *Service constraints*, which are the obligations the service should follow.

The figure below shows the adopted representation of a Service Object.

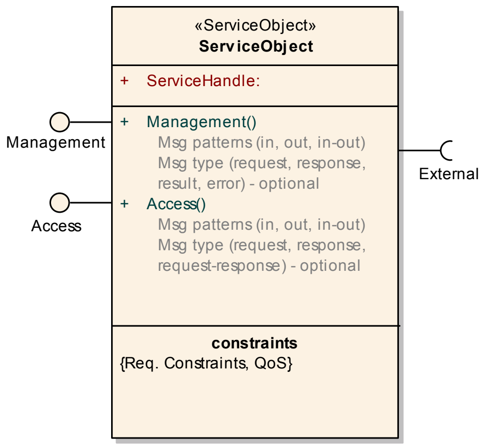
«ServiceObject»
ServiceObject
+ ServiceHandle:
+ Management()
Msg patterns (in, out, in-out)
Msg type (request, response, result, error) - optional
+ Access()
Msg patterns (in, out, in-out)
Msg type (request, response, request-response) - optional
constraints
{Req. Constraints, QoS}
Management
Access
External

Figure 3: Service Object Representation

The two provided interfaces (Access and Management) are characterized by the message patterns and type, included in the object definition. The interfaces can also be associated to information objects defined in the Information Viewpoint. This way, dependencies between Provided Interfaces and Information Objects may be created, enabling the establishment of links between the two most important Viewpoints of the reference architecture (Information and Computational).

On the other side, the External interface is not included in the object definition, since it consists in a reference to an interface provided by another service. This will be represented as an UML dependency.

Engineering Viewpoint

An engineering viewpoint focuses on the aggregation and deployment of services in Software Components, Nodes and Channels, which result of explicit physical and environmental constraints.

The G/S Engineering Viewpoint describes the way the functional entities, that were described in the Computational Viewpoint in terms of services, are located with respect to computing resources and network connections. Although the viewpoint is related with the physical distribution of services, it is technology independent, hiding the implementation aspects of the system.

Engineering Objects: the Engineering Viewpoint is expressed in terms of three basic Objects: the Component, the Node and the Channel.

The **Component** represents the logical implementation, in terms of aggregation and deployment, of one or more computational Services. It has the following attributes:

- *Core functionality*: service implementation; localization constrains; performance constraints; resources managed; infrastructure and support.

- *Management Interface*: set-up, resume and shutdown; service activation; logging and reporting; access and control.

- *Access Interface*: interface that exports the services implemented

- *External Interface*: interface to Channel Objects; interface to other components

The **Node** is the logical representation of a physical entity that has a set of computing and storage resources and is located at a specific premise. The node supports the activities of the Components that implement the functionality of the associated Service Objects. It has the following attributes:

- Core functionality: resources (interfaces, memory and disk storage); processing (CPU cycles); encapsulation (memory management and protection)

- Management Interface: component set-up

The **Channel** is the logical representation of networking associated services that establish the connection between the existing Nodes. A Channel contains specific behavioural, functional and physical attributes, related to the characteristics of the connection it represents. It has the following attributes:

- *Core functionality*: protocol implementation; link management (local, internetworking, space link, etc.)

- *Management Interface*: channel set-up; flow control

- *Binding Interface*: binding set-up; name resolution

In order to simplify the framework and improve the specification readability, the Engineering Object artefacts adopted use an UML simplified notation. The figure below shows the condensed representation of the three basic objects.

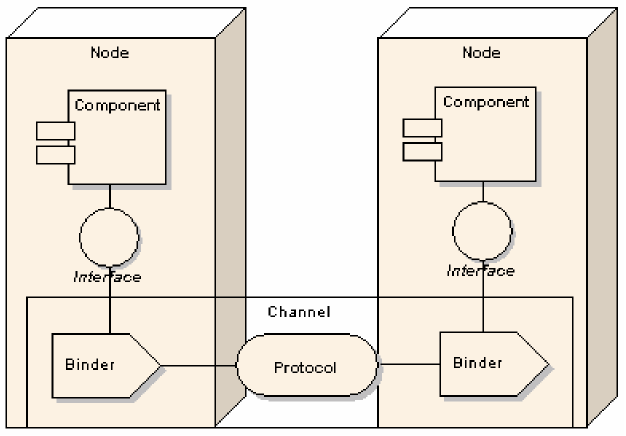

Figure 4: Representation of G/S Engineering Objects

Additional Accessory Views

Some specific views may be used to represent additional G/S aspects, like data flows, performance, RAMS (reliability, availability, maintainability and safety), etc… These views are derived from the previous ones, and may compose otherwise independent aspects or objects to depict situations that are not easily visible in any of the other views.

The Framework provides the means necessary to support the Reference Architecture specification. It enforces the RM-ODP standard and uses UML language to support the specification. The approach assumes some simplifications of the RM-ODP standard in order improve the readability and understanding of the UML artefacts.

The complexity of the Ground Systems demanded the usage of a tool to support the specification. Since there are no tools available that presently enforce the RM-ODP concepts, an off-the-shelf UML modelling tool was selected and customized accordingly.

3. Reference Architecture specification

Supported by a modelling tool, the reference architecture for Ground Systems Software has been specified, based on the Viewpoints and additional concepts described previously. The specification describes the whole ground segment, focusing particularly on the Operational Control System (OCS).

a) Enterprise Viewpoint

The specification of the Enterprise Viewpoint was mainly achieved through the identification of High Level Requirements for the Ground Systems.

During this process, characterization of the Ground Segment Systems based on ECSS-E70 [1] was developed, resulting in its high level hierarchical decomposition into G/S Elements, Systems, Subsystems and Logical Modules.

The figure below presents the logical representation of the G/S Enterprise Viewpoint, in terms of elements and systems.

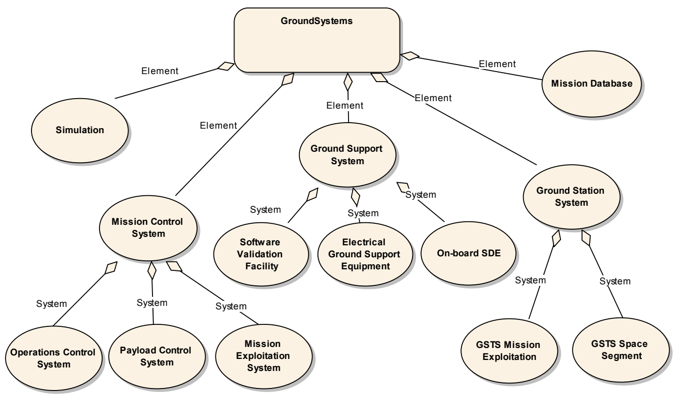

Figure 5 – Ground Segment decomposition model

The requirements uniquely identified and logically associated with each object, are loaded into the viewpoint and classified according to the hierarchical decomposition.

b) Information Viewpoint

The methodology followed to populate the Information Viewpoint was the following:

- The information flows between the ground system's functional entities are initially identified on the Data Flow View. This identification is achieved by means of a systematic analysis of the exchanges of information occurring between peer entities within each level of the hierarchical decomposition covering all systems, subsystems and logical modules of the Ground Segment.

- An Information Object is associated to each of the identified flows, that models its characteristics from the point of view of its structure and content type.

- A catalogue of information objects was finally developed, on which the logical relationships between all information objects identified were expressed, namely in terms of dependencies, similarities, composition and inheritance. A model representation of all the ground systems information entities was therefore obtained.

Data Flow View

The identification process described above is applied to all the levels of the hierarchical decomposition in order to identify the existing G/S systems information objects. Consequently, several dataflow views are obtained, each one providing a different level of abstraction.

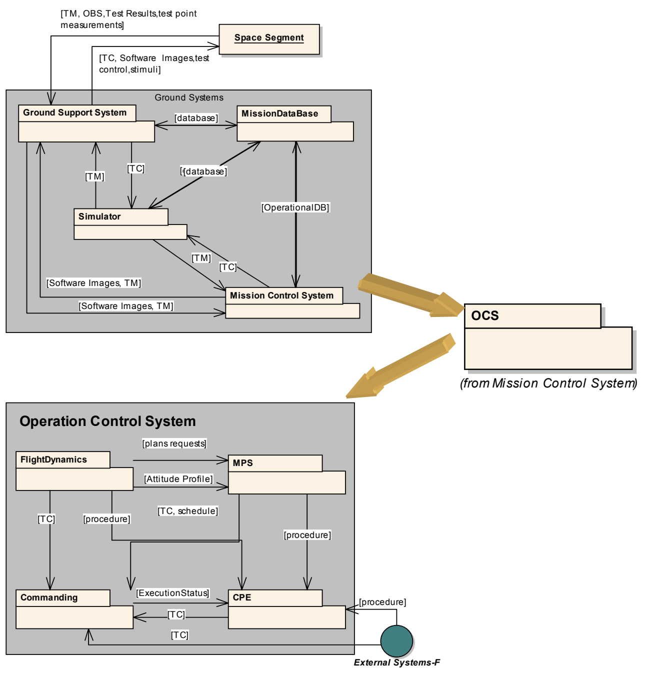

Figure 6 - Hierarchical levels of the Data Flow View

Figure 6 contains a simplified representation of this decomposition process.

The first level identifies the data flows existing between the G/S elements. The second level identifies the data flows existing between the subsystems that compose each G/S system (e.g. operation control system). The third level identifies the data flows between the logical modules that are part of each subsystem.

The Data Flow view is considered an accessory view within the framework and its specification is not formally linked to other views. However, it provides an excellent representation of the Systems, Subsystems and logical modules that compose the Ground Segment and assure the complete set of information objects has been identified.

Information Object Catalogue

As referenced previously, the flows identified in the Data Flow View are associated with Information Objects. Which are grouped in an Information Object Catalogue. This catalogue is organized to reflect the different Information Objects functional categories: Monitoring and Control Data Definitions (E-70-31), MCS Data Definitions, Storage, and Supporting Structures definitions.

Figure 7 depicts the Information Object Catalogue, with its different object categories, and an exploded view of an Information Object internal representation.

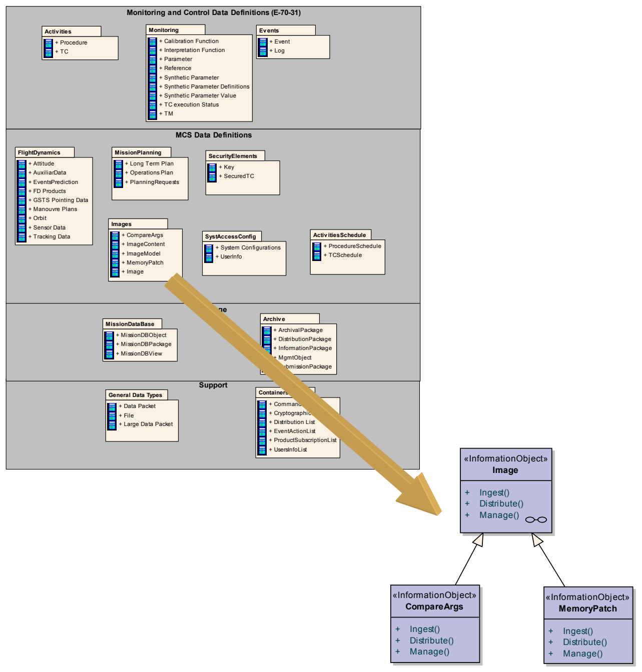

Figure 7 – Information Object Catalogue

As described in the architecture framework section, an Information Object is internally composed of two parts: the structure (ModelObject) and content (DataObject). The structure representation can be achieved using XML schema that can be used in further phases of the harmonization process to define the service interfaces (see Interface Specification section). The generation of such schema is independent from the organization of Ground Segment Systems and from their internal specificities.

c) Computational Viewpoint

While the information viewpoint is currently covering most Ground Segment systems, the computational viewpoint is mainly focused on the Operational Control System. The approach followed is bottom-up, which means that the services were initially identified based on low granularity functionalities, and on a second phase, the service interfaces exposed between subsystems were specified.

The identification of services uses the hierarchical decomposition and the specific G/S requirements to derive the behaviour of the entities and their interaction. Each logical module has an associated diagram, which contains the services that were identified for its representation. An example of the services defined for a module from the OBSM subsystem is shown in Figure 8.

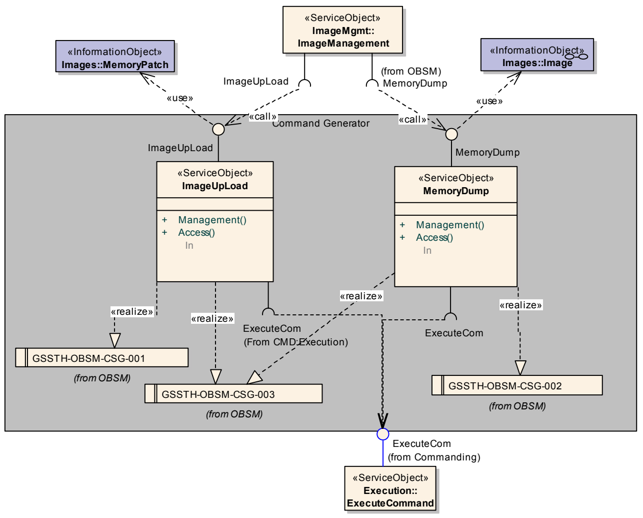

Figure 8 - Command Generator within OBSM subsystem

In Figure 8, two services are represented: one for Image uploading and another for Memory dumping. Each service is characterized by its attributes, namely responsibilities, constraints and interfaces (provided, required). The responsibilities and constraints are defined by links to specific requirements such as GSSTH-OBSM-CSG-001. The provided and required interfaces enable the representation of invocation relationships between services, and are also characterized with dependency links to the Information Objects defined in the Information Object Catalogue. The service "ImageManagment", external to the current logical module, invokes the ImageUpload service using the MemoryPatch information object as an argument.

Exposed Service Interfaces

After defining the services, it is important to identify the service interfaces existing between systems and subsystems. The first step is to detail the service invocations that occur between logical modules, as described in the figure below.

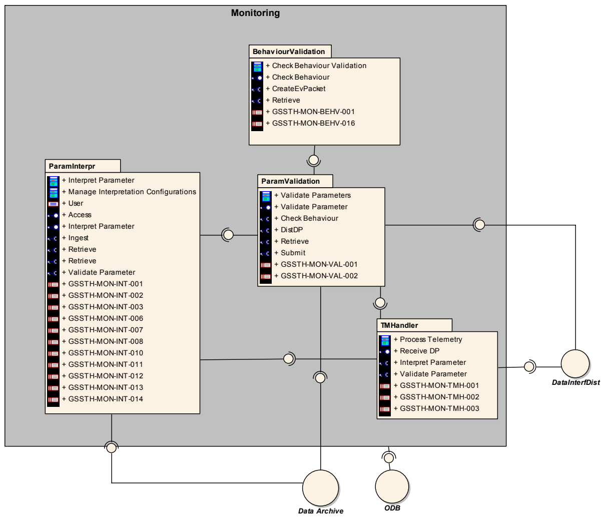

Figure 9 - Monitoring Computational View

Figure 9 represents the service invocations occurring between the logical modules that compose the Monitoring Subsystem. Modules are represented by UML packages. The icons on the packages represent the services that are part of each logical module, their provided and required interfaces, and their associated requirements.

After describing the service interfaces that connect logical modules, the process was reapplied at the Operational Control System level, with the consequent result of identifying the exposed service interfaces that allow service invocation between subsystems.

Figure 10 provides a simplified view of the Operational Control System exposed interfaces.

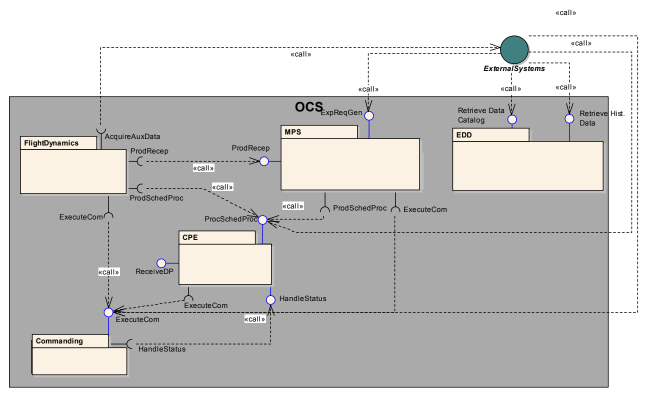

Figure 10 - OCS Computational view

The identification of the exposed interfaces and invocation relationships suggested a layered approach within the Operational Control System, as described in Figure 11:

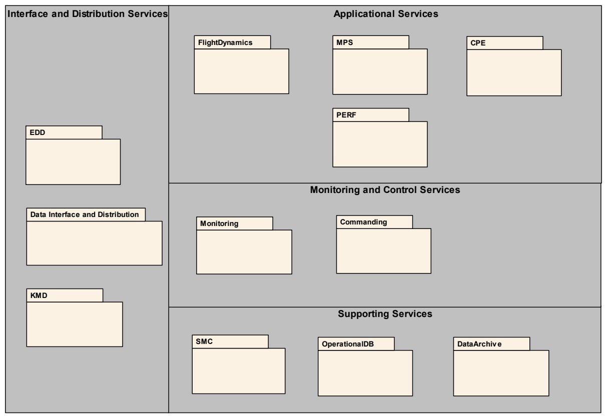

Figure 11 - OCS layering

Four layers were defined:

- The Application Services layer that implement application services. They invoke lower layer services, such as monitoring and control services, and supporting services.
- The Monitoring and Control Services layer, which invoke supporting services from the lower layer.
- The Supporting Services layer, which provide basic functionalities (such as Archiving) to the upper layers.
- The Interface and Distribution Layer, which is orthogonal to all the others, and provides a generic interface for distributing data among them.

With this characterization, it is possible to identify which subsystems implement high level services, and which implement supporting services. Such an approach improves system maintainability, since subsystems are considered black boxes. That is, their inner functionalities are encapsulated using a Service Oriented approach, and their main interfaces are exported to external world.

d) Engineering Viewpoint

The Engineering Viewpoint identifies specific resources, such as components and nodes where the services are executed, and communication channels that support the implementation of remote interactions.

The Engineering Viewpoint is composed of three related specific views, which are the Component, the Node and the Channel Views.

The methodology followed started by the definition of the Component View, which groups services into components, followed by assigning those components to nodes in the Node View, and finally by describing the necessary interconnection paths in the Channel View.

Figure 12 represents the steps followed.

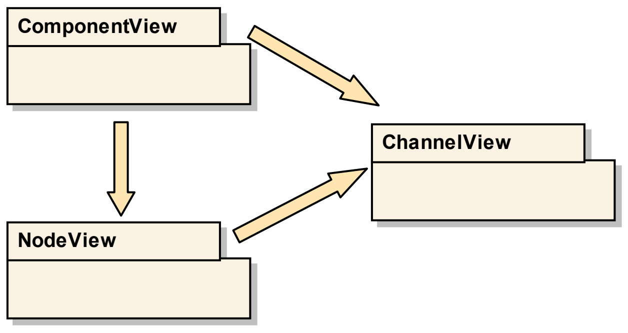

Figure 12 – Engineering Viewpoint approach

In each view, the specification started from the top level of the hierarchy, by introducing the entire Ground Segment context, and then detailing the Operational Control System.

The Engineering View also reflects the Service Oriented Architecture (SOA) model, which contains the following 3 elements: the Service Directory, the Service Consumer and the Service Provider. The Service Directory is where a Service Provider publishes its services, and where a service consumer discovers providers of a given service.

<u>Component View</u>

In this view, components that represent Service Directory elements, channels and OCS subsystems (e.g. Flight Dynamics, Archiving, Monitoring, Commanding) were identified.

A component is a composite object, in the sense that it may in turn be decomposed in sub-components. For instance components associated with the implementation of OCS subsystems contain sub-components that implement functionalities associated with the logical modules of each OCS subsystem.

Node View

Two levels of nodes were specified in this view: High-level nodes (Figure 13) represent facilities (OCC, GSTS), whilst Low-level nodes represent OCS processing units.

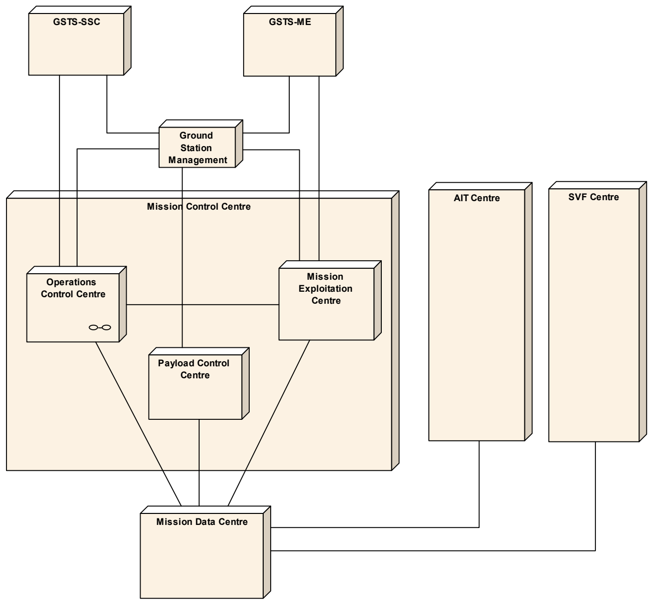

Figure 13 –Ground Systems Node View

In order to define the high-level nodes, the main processing centres were first identified: Mission Control Centre (MCC), AIT centre, etc.

The MCC was then decomposed according to E-70 into Operation Control Centre (OCC), Payload Control Centre (PCC) and Mission Exploitation Centre (MEC).

Finally, the OCC was split into nodes corresponding to groups of machines with a specific purpose, e.g. Flight Dynamics, Archiving, Monitoring and Control. These lower level nodes contain components, as identified in the Component View.

Additionally, each node includes a supporting component named "Service Agent", which allows service invocation and provisioning between nodes.

Channel View

This view combines the node view described above with the channel components that support communications. The components that implement the channel (interfaces, bindings, stubs, protocols…) are associated with the communicating nodes. The resulting view represents the establishment of communication links between high-level components in different nodes.

The Channel view also represents the communications existing between the service invocation and directory components. Each node contains a supporting component (the Service Agent), which communicates which its peers, and with the directory service through a dedicated Service Channel

The same approach applied in the Node View was followed for building the Channel View, with the representation of two levels. The first describes the communications between the high-level nodes of Ground Segment, and is represented in Figure 14. The second represents the communication between low-level nodes of OCC..

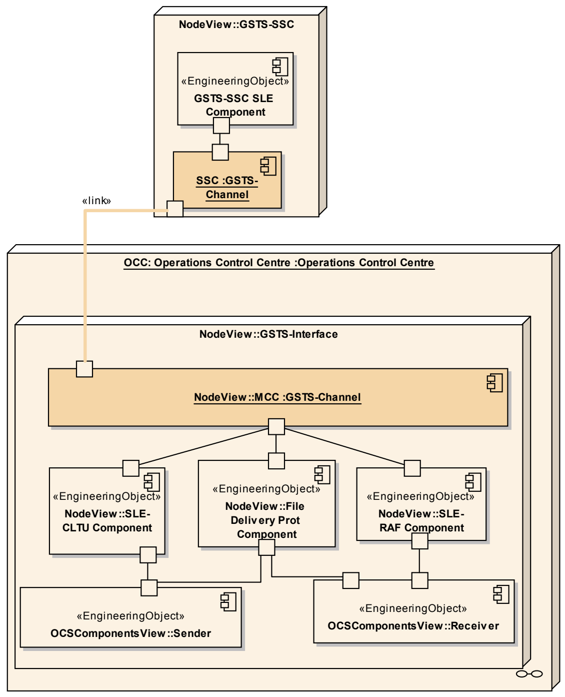
NodeView::GSTS-SSC
«EngineeringObject» GSTS-SSC SLE Component
SSC :GSTS-Channel
«link»
OCC: Operations Control Centre :Operations Control Centre
NodeView::GSTS-Interface
NodeView::MCC :GSTS-Channel
«EngineeringObject» NodeView::SLE-CLTU Component
«EngineeringObject» NodeView::File Delivery Prot Component
«EngineeringObject» NodeView::SLE-RAF Component
«EngineeringObject» OCSComponentsView::Sender
«EngineeringObject» OCSComponentsView::Receiver

Figure 14 – OCS Channel View (simplified)

At the G/S level, 3 channels were defined, based on their functional purpose:[5]

[5] These channels constitute the Ground Communications Subnet (GCS) referred in ECSS E-70.

- Ground Station channel - connects the mission control centres with the ground stations

- GSTS management - connects all the nodes involved in the management of the ground stations

- MDB channel - connects all processing centres with the central Mission Database.

At the OCC level, 3 channels were also defined, based on the requirements existing for the underlying physical links, according to the needs of the existing services:

- Service channel – this is a general-purpose channel, with no particular latency or bandwidth requirements on the physical link. It is used mainly to setup communication links and for service invocation.

- Distribution channel - is characterised by a very low latency and high speed, as it is intended for (near) real-time communications. It is used mainly for distribution of packets (TM, TC, EV) within the OCS system, as well as communication with the ODB.

- Archive channel - offers a large bandwidth, in order to support the transfer of large sets of data. It is targeted for archiving services.

4. Interface specification approach

In a service oriented architecture the interface is the key component of the interaction between services. It defines the operations (methods), data structures of parameters and results, and the behaviour of the interaction. The interface key elements are depicted in Figure 15:

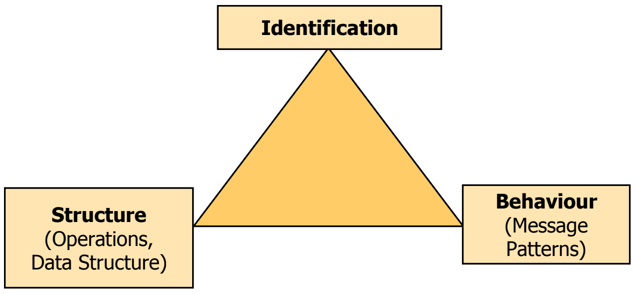
Identification
Structure (Operations, Data Structure)
Behaviour (Message Patterns)

Figure 15 - Interface key elements

The next harmonization step is to initiate the specification of the most relevant Ground Segment interfaces. The approach to be followed may use the elements described above, integrated with the Reference Architecture. In the Computational Viewpoint, the service interfaces support the specification of interface operations. The Information Viewpoint allows the representation of the data structures.

Based on this specification, different technologies may be used to support the interface description. The WSDL (Web Services Definition Language) and IDL (Interface Definition Language) are the most promising ones.

In terms of structure, WSDL has the necessary richness for correctly achieving the specification of the elements identified above. It provides tags to define the operations, parameters and results of operation invocation. Using this Definition Language based on XML, the Information Objects views may be used straightforward for the definition of arguments and results. Furthermore, WSDL allows the definition of the service binding to particular transport protocols (e.g. HTTP, SOAP) and a way for universally specifying the identification and location of services (URI). The specification of the interface behaviour can be achieved according to W3C message patterns schemas.

5. Conclusions

The Reference Architecture, as well as the Framework presented in this paper, brings important benefits to the harmonization process. They introduce an innovative approach for describing the Ground Segment software systems based on a Service Oriented Architecture approach, which allows higher degrees of decoupling and independence between its main components.

The architectural viewpoints introduced also enable a formal representation of the Ground Segment systems, thus providing a more precise description of their detailed functionalities and interactions, and allowing for a consolidation of the terminology used in the domain.

The usage of a UML tool for implementing the specification also provides several advantages: it enforces the model's coherency; provides for a user-friendly navigation experience across the different viewpoints; allows the automatic generation of documents; and creates traceability between objects, both within and across viewpoints.

All these factors are of great value with respect to improving the architecture's readability, comprehension, and correctness.

As for systems interoperability and maintainability, the benefits of this approach are even greater. The definition of "standard" (sub) systems in terms of a Service Oriented Architecture (SOA), on which internal specific functionality is encapsulated in external well defined interfaces, is the key factor for decoupling otherwise monolithic applications.

Using this approach, the overall G/S systems functionality can be built upon sets of pre-existing service components that are assembled in-line with mission needs, according to its specific business requirements.

The usage of several COTS solutions from different providers also becomes much easier, as long as each component respects the reference architecture.

The compliance with a reference architecture also means that each system may become more self contained and generic, thus reducing the need of additional developments for each specific mission.

The adoption of the reference architecture may also trigger quality improvements for existing and future systems. The modularity of the approach, together with the clear distinction between functionality (computational), data (information) and deployment (engineering), greatly improves the maintainability and the portability of systems. New technologies can be used and new platforms can be supported with practically no need for changes either in the information or computational Viewpoints.

Future developments and enhancements to this architecture include the specification of interfaces between key components, which will further improve interoperability, and which may be used to enforce the compliance of new products with the reference architecture.

REFERENCES

[1] ECSS E-70 - Ground systems and operations - Part 1: Principles and requirements, 25/04/2000
[2] RM-ODP Open Distributed Processing Reference Model: ISO/IEC 10746 Part 1: Overview; ISO/IEC 10746 Part 2: Foundations; ISO/IEC 10746 Part 3: Architecture
[3] CCSDS RASDS-07 - Reference Architecture for Space Data Systems - Consultative Committee for Space Data Systems Architecture Working Group, Issue 0.7, March 11, 2003.
[4] UML Profile for Enterprise distributed Object Computing (EDOC): OMG Enterprise Collaboration Architecture (ECA), v1.0; OMG UML Profile for Patterns, v1.0.
[5] ECSS E-40 - Space engineering, ECSS-E-40B Draft 1, 15 February 2002.
[6] Monitoring & Control Data Access (RFP #2) Request For Proposal OMG Document, 2002-01-04.

BIOGRAPHY

Nestor Peccia *is the Head of Data System Infrastructure Division, Ground Engineering Department at the European Space Agency (ESA). He has developed and led development of Mission Control Systems and Ground Segment Infrastructure software during the last 18 years. He is also Chairman of the European Technology Harmonisation on Ground Software Systems and Area Director of the CCSDS Mission Operations and Information Management Services. He has a MEng from Buenos Aires University, Argentina.*

Nuno Duro *is the head of the Ground Segment Division of Critical Software. He has leaded the engineering support project which developed the reference architecture presented in this paper. He has also been the Project Manager of various activities related to Ground Segment software development, especially on field of MCS and data processing. He participated in ISVV of Cryosat on-board software and on the development dynamic software verification methods and other RAMS techniques. Prior to Critical Software, he has worked in data communications on private networks and multimedia information kiosks. He has a graduate degree in Computer Science from the University of Coimbra.*

Flávio Moreira *is a project manager for the Space Business Unit of Critical Software, in Portugal. He has led development of software for mission control systems and for analysis of science data, and participated in the definition of the reference architecture for the ground segment. Previously he worked at Alcatel Portugal, focusing on architectural issues of distributed systems for the telecommunications business. He has a graduate degree in Computer Sciences, and a post-graduate degree in Information Systems and Technology.*

José Rogado *has been working as a senior Consultant in Information Technologies and Systems Architecture with Critical Software. Prior to Critical Software, his work assignments include working as a Technical Manager (for a Telecom Mobile Operator), as a Technology Consulting Manager (for Cap Gemini Ernst & Young Portugal), and as Business Unit Manager (for Link Consulting). Relevant previous assignments in other countries include positions as Senior Research Engineer, at the Open Software Foundation Research Institute, Boston and Grenoble (7 years), and as an Invited Researcher at INRIA, Paris (4 years). He holds Ph.D. and a Ms.Sc. in Computer Science from the Paris VI University, and graduated as an Electronic Engineer at the Lisbon Polytechnic Institute.*

José Reis *is a project engineer for the Space Business Unit of Critical Software, in Portugal. He has participated in the development of software for mission control systems and for analysis of science data, and in the* *definition of the reference architecture for the ground*

segment. Previously he worked at X-Cago B.V. Netherlands, focusing on development of web applications. He has a graduate degree in Computer Engineering at the Lisbon Polytechnic Institute (Instituto Superior Técnico).