



**Universitat Oberta
de Catalunya**

www.uoc.edu

FINAL DISSERTATION

COMPUTER ENGINEERING STUDIES 2013/2014

«A DESIGN FOR AN ADVANCED ARCHITECTURE OF SATELLITE GROUND SEGMENTS»

BY RAMOS PÉREZ, JOSÉ JULIO

ABSTRACT

According to (Euroconsult, 2008), the number of new satellites launched through 2017 will double the launches over the previous ten years. The world manufacturing market for EO satellites will grow from \$14.8 billion to \$16.9 billion, a 14% increase. This growth will be fuelled primarily by the promising private sector and dynamic emerging space programs after the established programs (such as NASA, ESA, CNES, ISRO, etc.) that have historically dominated the Earth Observation sector, have started backing out of this sector. They will remain the leading actors in terms of number of satellites to be launched but their share of total satellites will plummet from 77% to a mere 36%. This means that in three years, the 64% of new EO satellite developments will be curated by newcomers in the EO sector, with different business objectives and strategies than the traditional customers. New actors in the business even include established multinationals in the Internet and information sectors like Google, which has just acquired an EO satellite company, Skybox Imaging, for acquiring and exploiting Very-High resolution optical images (SkyBox Imaging, 2014) to enrich their data services.

The traditional satellite upstream sector must adapt to all those new conditions if it wants to remain noteworthy but first it must overcome several problems present in one of the high-level systems of every satellite mission, the Ground Segment.

Ground Segment developments present nowadays a series of problems including:

- Excess of conservatism - Use of obsolete, outdated, or in the best case scenario, technologies out-classed by current state-of-the-art techs already operational and thoroughly tested in much more competitive markets like finance, telecommunications or Internet;
- Lack of an integral approach to the systems design - Non-optimal distribution of work to 'vertical' teams in charge of developing complete subsystems force subcontractors to use a diverse team of specialists in many different areas (applications, network, systems, etc.) that would eventually lead to reach different solutions to common problems present in the rest of subsystems;
- Short-sighted developments - Every new ground system development is requirement-driven, focused in complying with present project needs but without a wider, longer-term strategy. This way every team starts relying solely in the talent and experience gathered by the main contractor and cannot take use of past best-practices.

This final dissertation proposes <<a design for an advanced architecture of Satellite Ground Segments>> with the objective to mitigate the aforementioned problems and cope with the imminent changes in the EO business sector. This design is focused in supporting an undefined generic Earth Observation satellite with a minimal set of functionalities, on a minimal deployment of capabilities. Although this system is only described during nominal operational phases it would be also able to support the rest of operational phases like Launch and Early Orbit Phase (LEOP) or decommissioning.

At the end, it will be shown that the final goals of this project are met, that is, to reduce development costs, reduce the time to market of a mission concept, improve the quality of resulting systems, and generally to incentive positive changes and renewed dynamism to the EO satellite sector for ultimately closing the gap between the space-given benefits of satellites and the public in general.

TABLE OF CONTENTS

1 INTRODUCTION	6
1.1 PROJECT DESCRIPTION	8
1.2 OBJECTIVES	9
2 PROJECT	10
2.1 INTRODUCTION	11
2.1.1 PURPOSE	11
2.1.2 SCOPE	11
2.2 ENTERPRISE VIEWPOINT	12
2.2.1 USER REQUIREMENTS	12
2.2.2 FLIGHT OPERATIONS	13
2.2.3 PRODUCT DATA GENERATION	14
2.2.4 PRODUCT DISSEMINATION	14
2.2.5 ENTERPRISE MANAGEMENT	15
2.2.6 NON-FUNCTIONAL REQUIREMENTS	18
2.3 INFORMATION VIEWPOINT	21
2.4 COMPUTATIONAL VIEWPOINT	24
2.4.1 USER INTERFACES LAYER	27
2.4.2 APPLICATIONS LAYER	28
2.4.3 SERVICE INTEGRATION LAYER	31
2.4.4 SERVICES LAYER	33
2.4.5 DATA INTEGRATION LAYER	36
2.4.6 DATA LAYER	37
2.4.7 PLATFORM LAYER	39
2.4.8 INFRASTRUCTURE LAYER	44

2.4.9 SECURITY SERVICES LAYER	45
2.5 ENGINEERING VIEWPOINT	47
2.5.1 DATA PROCESSING	47
2.5.2 SATELLITE CONTROL	48
2.6 TECHNOLOGY VIEWPOINT	49
2.6.1 SOFTWARE	49
2.6.2 NETWORK	57
2.6.3 HARDWARE	60
3 CONCLUSIONS	62
4 GLOSSARY	63
5 BIBLIOGRAPHY	65

TABLE OF ILLUSTRATIONS

ILLUSTRATION 1: USER REQUIREMENTS STRUCTURE	12
ILLUSTRATION 2: GROUND SEGMENT DATA MODEL	22
ILLUSTRATION 3: FUNCTIONAL DECOMPOSITION	26
ILLUSTRATION 4: USER INTERFACES COMPONENTS	27
ILLUSTRATION 5: APPLICATIONS LAYER COMPONENTS	28
ILLUSTRATION 6: SERVICE INTEGRATION LAYER COMPONENTS	31
ILLUSTRATION 7: SERVICES LAYER COMPONENTS	33
ILLUSTRATION 8: DATA INTEGRATION LAYER COMPONENTS	36
ILLUSTRATION 9: DATA LAYER COMPONENTS	37
ILLUSTRATION 10: PLATFORM LAYER COMPONENTS	39
ILLUSTRATION 11: INFRASTRUCTURE LAYER COMPONENTS	44
ILLUSTRATION 12: SECURITY SERVICES LAYER COMPONENTS	45
ILLUSTRATION 13: NETWORK DIAGRAM	58
ILLUSTRATION 14: NETWORK DETAILED DIAGRAM	59
ILLUSTRATION 15: PHYSICAL DEPLOYMENT	61

1 INTRODUCTION

This section gives a justification of the work, its objectives and methodology taken, as well as a description of the structure of this document.

This project provides <<a design for an advanced architecture of Satellite Ground Segments>> trying to solve the following problems usually found in the field:

- Excess of conservatism - Use of obsolete, outdated, or in the best case scenario, technologies out-classed by current state-of-the-art techs already operational and thoroughly tested in much more competitive markets like finance, telecommunications or Internet;
- Lack of an integral approach to the systems design - Non-optimal distribution of work to 'vertical' teams in charge of developing complete subsystems force subcontractors to use a diverse team of specialists in many different areas (applications, network, systems, etc.) that would eventually lead to reach different solutions to common problems present in the rest of subsystems;
- Short-sighted design - Every new ground system development is requirement-driven, focused in complying with present project needs but without a wider, longer-term strategy. This way every team starts relying solely in the talent and experience gathered by the main contractor and cannot take use of past best-practices.

Therefore, the objectives and goals of this system are:

- To update essential systems in a highly technological niche sector, that is currently living a world-wide revolution in the use of its products and services, but urgently needs to catch up with state-of-the-art technologies, already present in other sectors;
- To enhance the classical production cycle by identifying possible commonalities in the design, thus avoiding repetition of functional and technological items. This would lead to a more cost-efficient, quicker and robust development;
- To enable cost and risk reductions by sharing development, maintenance and operations of a single, common infrastructure as well as the exchange of functional components across deployments and teams;
- To give a reference architecture framework for Earth Observation ground systems. This way, future developments could take advantage of the design and ideas shown in this project and adapt them for the peculiarities of their missions.

The methodology followed for describing this system is the Reference Architecture for Space Data Systems defined by the Consultative Centre for Space Data System (CCSDS, 2008) and their Reference Architecture for Space Information Management (CCSDS, 2013). The high-level requirements are a selected extract of the Geostationary Operational Environmental Satellite (GOES) Level I Requirements definition (NASA, 2012).

It is worth noting that there is already an initiative organized by the European Space Agency (ESA) with similar objectives. The European Ground Systems Common Core (EGS-CC) would provide a common infrastructure to support the space systems monitoring and control in pre- and post-launch phases. The differences with this project reside in while that program is focused on high-level functionalities, this

final dissertation does not explore the details of those functions but the “Enterprise Management” functionality instead. It focuses on the “data-centre” aspect of the system rather than in the “business” side. The author declares that he came up with the original idea of this project and started the redaction of this final dissertation unacquainted of its existence but, after knowing about its details almost close to finishing the document, he has used some recommendations of the EGS-CC available documentation in the “Technology Viewpoint” chapter. Nevertheless, the existence of this multinational, multiagency, multimillion top-level programme shows that the key players in the industry has identified the same problems and adopted similar approaches to solve them, albeit in much more detail and with much more resources and budget. This seems to validate the author’s premises, legitimate its objectives and approaches and approve some of the solutions he is proposing.

This document is structured in the following sections:

- Section 1, Introduction – This section, which provides a justification of the work, its objectives and methodology taken.
- Section 2, Project – This section is the core of the document, providing a complete description of the work made.
- Section 3, Conclusions – This section lists the extracted conclusions of this work and possible future evolutions and lines of work.
- Section 4, Glossary
- Section 5, Bibliography

1.1 PROJECT DESCRIPTION

This project proposes a preliminary architectural design for a control and data processing center, also known as 'ground segment', for Earth Observation satellites.

This design is focused in supporting the operations of an undefined generic Earth Observation satellite with a minimal set of functionalities, on a minimal deployment of capabilities. Although this system is only described during nominal operational phases it would be also able to support the rest of operational phases like Launch and Early Orbit Phase (LEOP) and decommissioning.

1.2 OBJECTIVES

Section 1 has already identified several problems in the current status of the Ground Segment field.

This project has the following targets aimed to diminish those problems:

- To update essential systems in a highly technological niche sector, that is currently living a world-wide revolution in the use of its products and services, but urgently needs to catch up with state-of-the-art technologies, already present in other sectors;
- To enhance the classical production cycle by identifying possible commonalities in the design, thus avoiding repetition of functional and technological items. This would lead to a more cost-efficient, quicker and robust development;
- To enable cost and risk reductions by sharing development, maintenance and operations of a single, common infrastructure as well as the exchange of functional components across deployments and teams;
- To give a reference architecture framework for Earth Observation ground systems. This way, future developments could take advantage of the design and ideas shown in this project and adapt them for the peculiarities of their missions.

The final goals of this project are then, to reduce development costs, reduce the time to market of a mission concept, improve the quality of resulting systems, and generally to incentive positive changes and renewed dynamism to the EO satellite sector for ultimately closing the gap between the space-given benefits of satellites and the public in general.

2 PROJECT

This section is the core part of the document as it provides a complete description of the work made.

This architectural design will follow the CCSDS Reference Architecture (CCSDS, 2008) that mainly adapts the RM-ODP¹ enterprise architecture framework for describing space data system, using these five defined model viewpoints:

- The enterprise viewpoint, which focuses on the purpose, scope and policies for the system. This project will include a user requirements section describing what functionalities need to be made available.
- The information viewpoint, which focuses on the data and information processing that will be performed. This project will include a section to describe the information model including supporting data.
- The computational viewpoint, which decomposes the system functionality into objects which interacts at interfaces. This project will include a section describing the system functionality and its decomposition.
- The engineering viewpoint, which focuses on the mechanisms and functions required to support the dynamic interactions between distributed objects in the system. This project will include a section describing some operational cases performed by the system to manage the information and provide the functionality.
- The technology viewpoint, which focuses on the final choice of technology of the system. This project will include a section listing the technologies chosen to provide the processing, functionality and presentation of information.

This project does not include the communications viewpoint, as it considers it out of scope.

The system's high-level functionality presents the following characteristics focused on enabling the aforementioned project's objectives:

- Service oriented, component based, layered architecture;
- Use of open components and standard interfaces;
- Choice of isolated but cross-compatible technologies;
- High Availability, High Performance, High Throughput and Scalable computational infrastructure;
- Long term maintainability by seamless adaptation to new technologies.

¹ Also known as ISO/IEC 10746, as defined by the International Organization for Standardization (ISO), the International Electro technical Commission (IEC) and the Telecommunication Standardization Sector (ITU-T).

2.1 INTRODUCTION

2.1.1 PURPOSE

These sections provide the functional requirements and constraints declared by a customer for developing a ground segment concept for earth observation missions, the description of data being managed and treated, the functionality finally being offered by the system design and the technologies proposed to reach that design.

2.1.2 SCOPE

The scope of the requirements used to describe the generic EO mission reflect current business needs in the space sector obtained from an informal study of a NASA mission (NASA, 2012) and incorporates the author's personal experience in the sector.

This project will not detail functionality of space-related applications (Flight Operations -FO-, Payload Data Generation -PDG- and Product Dissemination -PD-) but will focus on Enterprise Management (EM) functions and general requirements instead.

This project does not consider within its scope the following aspects:

- Details of the High-level functionalities of a Ground Segment, i.e., does not describe the details of a mission planning system or data processors;
- System development life cycle. The project considers only one operational platform: no development, testing, validation, or pre-operational infrastructures are included in the system. The system is also considered "accepted" by the customer, although it does not describe any validation/acceptance process.
- Operational, maintenance, contingency plans. This project does not cover operational concepts performed by human staff, even though they should be part of a more complete design.
- CCSDS Communication viewpoint and Satellite communication protocols. The author of this project is not capable to provide an alternative, better solution for those areas of the Ground Segment.
- System sizing exercises describing the data acquisition or communication budgets, data retention policies, dissemination demand or performance metrics. Therefore, it is not able to give definitive characteristics on the network or rest of hardware devices, including those related with cooling and data-center security and safety.
- All data transmissions are assumed to be secure. No additional discussions on security are given in this project for confidentiality reasons.

2.2 ENTERPRISE VIEWPOINT

This section describes the system from the enterprise viewpoint, which focuses on the purpose, scope and policies for the system. It describes the business requirements and how to meet them.

The system can be described by listing the set of requirements given by the customer (user requirements) and which it should meet at the end.

The user requirements are grouped in the hierarchy shown in Illustration 1.

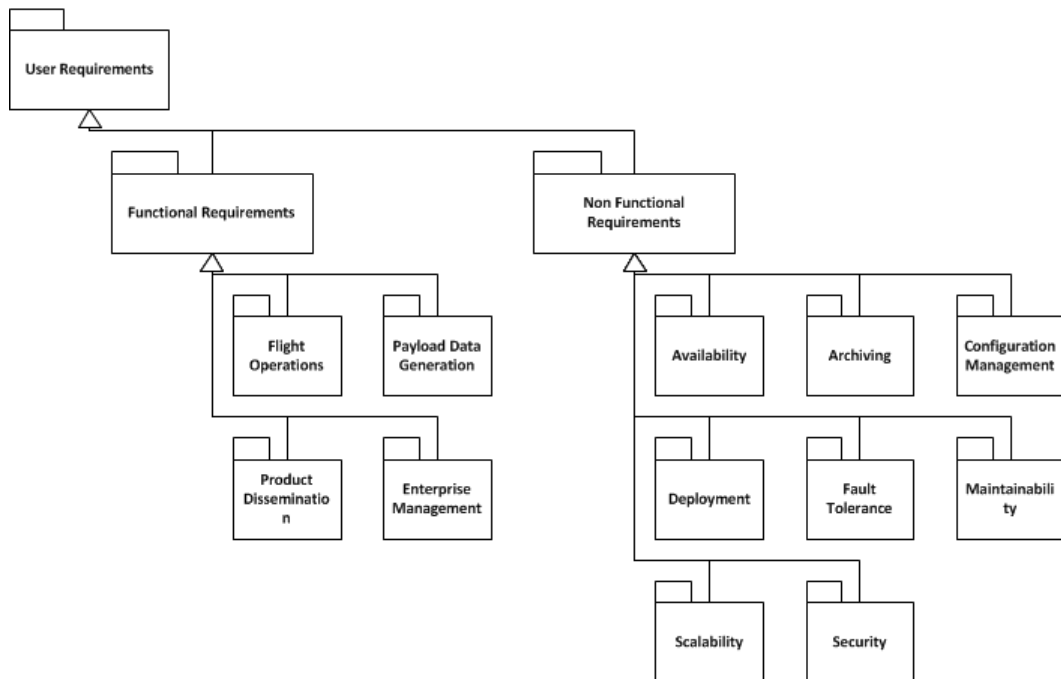


Illustration 1: User requirements structure

2.2.1 USER REQUIREMENTS

OBJECTIVES

SR-1.1.1.1	Support to mission operations	The Ground Segment shall support the primary mission operations goals to safely launch and operate the EO spacecraft.
SR-1.1.1.2	Generation of mission data products	The Ground Segment shall generate all data products defined for this EO mission.
SR-1.1.1.3	Mission data access	The Ground Segment shall provide access to those data products to the EO user community.
SR-1.1.1.4	Design drivers	The Ground Segment shall be designed,

		<p>developed, integrated and tested in a manner minimizing the overall program costs by using:</p> <ul style="list-style-type: none"> • Open standards • Modular, scalable, flexible system design • When possible, Commercial Off-The-Shelf (COTS) products that are commercially available and can be bought “as is”.
--	--	--

FUNCTIONS

SR-1.1.1.5	Ground Segment functions	<p>The Ground Segment shall provide the following functions:</p> <ul style="list-style-type: none"> • FO – Flight Operations. Including satellite mission operations (telemetry and command), mission scheduling, mission state-of-health and satellite orbital analysis. • PDG – Product Data Generation. Including collecting raw data from the spacecraft, and processing it to Level 0 and up to Level 1b operational data. Creation of higher level data products (Level 2/2+) • PD – Product Dissemination. Including distribution and access to data and formatted products to the user community. • EM – Enterprise Management provides layers of common resources to the rest of the Ground Segment and supports all operational functions by monitoring, assessing, and controlling the configuration of the operational systems, networks, and communications for the Ground Segment.
------------	--------------------------	--

2.2.2 FLIGHT OPERATIONS

OVERVIEW

Flight Operations (FO) encompasses all operational functions of the spacecraft and instruments.

FUNCTIONS

SR-1.1.1.6	Support to Space Segment operations	The Ground Segment shall provide terrestrial interface components to support all Space Segment operational phases.
SR-1.1.1.7	Communications with Space Segment	The Ground Segment shall perform engineering analysis on telemetry, command and event data for the life of the mission.
SR-1.1.1.8	System performance	The Ground Segment shall collect and report metrics related to system performance.

2.2.3 PRODUCT DATA GENERATION

OVERVIEW

The Payload Data Generation (PDG) function creates Level 1b and Level 2+ products for the full EO core product set.

FUNCTIONS

SR-1.1.1.9	Product generation	The Ground Segment shall accept RAW level-0 products, process Level-1 products and derive Level-2+ products.
SR-1.1.1.10	Product storage	The Ground Segment shall store all data required to reproduce the full production of EO series products (all Level 1b, Level 2, and Level 2+ products) for 7 days from all input data (Level 0 data), auxiliary and metadata.
SR-1.1.1.11	Product quality	The Ground Segment shall maintain the quality of all products.

2.2.4 PRODUCT DISSEMINATION

OVERVIEW

The Product Distribution (PD) function provides distribution of Level 0, Level 1b, Level 2+, and associated mission data produced by PDGS.

FUNCTIONS

SR-1.1.1.12	Access to products data	The PD shall provide user access to all generated EO
-------------	-------------------------	--

		data products.
SR-1.1.1.13	Products dissemination	The PD shall send L1b data, L2+ data, and associated metadata to the EO data portal users.
SR-1.1.1.14	Products archiving	The PD shall archive 7 days of data and products after product generation for redistribution.

2.2.5 ENTERPRISE MANAGEMENT

OVERVIEW

Enterprise Management (EM) provides layers of common resources to the rest of the Ground Segment and supports all operational functions by supervising the elements that comprise the operational systems and networks for the EO Ground Segment. In the EM context, those common resources supervision is defined as the ability to monitor, report, and provide capability for an operator response to anomalous conditions. EM functions underpin the services, platform and infrastructure that links the FOS, PDGS, and PDS functions and supports automation. While direct control of various systems may be implemented within the individual elements, EM provides a high-level layer of supervision over the end-to-end GS.

GS operators will have access to the EM functionality for insight to their local site and to the distributed GS components, infrastructure, and interfaces. As the EM functionality receives status and other information provided by the distributed GS functions, operators would be able to monitor, trend, and perform other supervisory activities.

The EM function will be a primary tool of real time operators and engineers to provide greater operational availability, efficiency, and safety of the EO Ground Segment system.

LAYERS

SR-1.1.1.15	Layering	<p>The EM shall provide the following layers of common resources to the rest of the Ground Segment functions:</p> <ul style="list-style-type: none"> • User Interface Layer; • Applications Layer; • Service Integration Layer; • Services Layer; • Data Integration Layer; • Data Layer; • Platform Layer;
-------------	----------	--

		<ul style="list-style-type: none"> • Infrastructure Layer; • Security Services Layer.
--	--	---

USER INTERFACE LAYER FUNCTIONS

SR-1.1.1.16	Interface to satellite	The EM shall supervise GS communications with the EO satellite.
SR-1.1.1.17	Product distribution	The EM shall supervise Product Distribution to the EO user community.
SR-1.1.1.18	Product portal	The EM shall provide a Product Portal function for accessing EO products.

APPLICATION LAYER FUNCTIONS

SR-1.1.1.19	Application functions	The EM shall support the FO, PDG and PD functions.
-------------	-----------------------	--

SERVICE INTEGRATION LAYER FUNCTIONS

SR-1.1.1.20	Messaging	The EM shall provide a common messaging function to communicate distributed applications.
SR-1.1.1.21	Common interface to services	The EM shall provide a common interface with functions on the underlying Services Layer.

SERVICES LAYER FUNCTIONS

SR-1.1.1.22	Web server	The EM shall provide a web server function accessible to the applications layer.
SR-1.1.1.23	Application server	The EM shall provide an applications server function as a common environment where application functions can run.
SR-1.1.1.24	Processing server	The EM shall provide a common processing server to run large scale processing of product data sets.

DATA INTEGRATION LAYER FUNCTIONS

SR-1.1.1.25	Database services	The EM shall provide database services to be accessed by the common Services layer.
-------------	-------------------	---

SR-1.1.1.26	File systems access	The EM shall provide 'file system' accessibility to data sets.
-------------	---------------------	--

DATA LAYER FUNCTION

SR-1.1.1.27	Access to operational data	The EM shall provide access, via the Data Integration Layer, to operational data sets.
SR-1.1.1.28	Access to products data	The EM shall provide access, via the Data Integration Layer, to product data sets.
SR-1.1.1.29	Access to enterprise data	The EM shall provide access, via the Data Integration Layer, to enterprise data sets.

PLATFORM LAYER FUNCTION

SR-1.1.1.30	Systems monitoring	The EM shall monitor and report the end-to-end status and performance of all GS system elements (hardware and software), networks, and communication links and antennae operations.
SR-1.1.1.31	Real time monitoring	The EM shall provide real time textual and graphical display of system performance and status.
SR-1.1.1.32	Management of networks and external interfaces	The EM shall supervise all GS networks and interfaces to external systems.
SR-1.1.1.33	Performance monitoring	The EM shall provide support to network and system performance metering.
SR-1.1.1.34	Consistent Operating Systems	The EM shall provide a consistent set of Operating Systems for supporting the rest of software systems.
SR-1.1.1.35	Distributed file system	The EM shall provide a distributed file system for storing software and data sets.

INFRASTRUCTURE LAYER FUNCTION

SR-1.1.1.36	Networking	The EM shall provide all hardware and software devices
-------------	------------	--

		associated with the GS networks (LAN and WAN).
SR-1.1.1.37	Server machines	The EM shall provide all GS server machines.
SR-1.1.1.38	Client devices	The EM shall provide all GS client devices.

SECURITY SERVICES LAYER FUNCTION

SR-1.1.1.39	Enterprise IT security	The EM shall supervise IT enterprise security.
SR-1.1.1.40	Authentication	The EM shall provide authentication services to all GS functions.
SR-1.1.1.41	Authorization	The EM shall provide authorization services to all GS functions.

2.2.6 NON-FUNCTIONAL REQUIREMENTS

AVAILABILITY

SR-1.1.1.42	Downtime	The Ground Segment availability shall be at least 99.999% over the system lifetime. Availability is defined as the fraction of time the ground segment has full functionality over a monthly interval.
SR-1.1.1.43	Mean Time To Restore	The Ground Segment mean time to restore functionality shall be less than 2 hours.

ARCHIVING

SR-1.1.1.44	Software archives	The Ground Segment shall archive all operational software versions for the life of the EO mission.
SR-1.1.1.45	Mission operations archive	The Ground Segment shall archive data supporting mission operations for the life of the EO mission.
SR-1.1.1.46	Product performance archive	The Ground Segment shall archive data supporting product performance evaluation for the life of the EO mission.

CONFIGURATION MANAGEMENT

SR-1.1.1.47	Configuration identification	The Ground Segment shall set and maintain baselines, which define the system at any point in time.
SR-1.1.1.48	Configuration control	The Ground Segment shall manage all changes on configuration items of the system.
SR-1.1.1.49	Configuration accounting	The Ground Segment shall record and report configuration item descriptions and all departures from the baseline.
SR-1.1.1.50	Configuration restoration	The Ground Segment shall be able to restore past system configurations.

DEPLOYMENT

SR-1.1.1.51	Automatic deployment	The GS shall automate deployment (installation and configuration) activities for all software components.
-------------	----------------------	---

FAULT TOLERANCE

SR-1.1.1.52	Fault detection	The Ground Segment shall perform fault detection and isolation, i.e. identify when a fault has occurred, and pinpointing the type of fault and its location.
SR-1.1.1.53	No single point of failure	The Ground Segment shall continue to operate without interruptions upon the failure of one of its components.
SR-1.1.1.54	Fault containment	The Ground Segment shall prevent propagation of failures from one component to the rest of the system.

MAINTAINABILITY

SR-1.1.1.55	Maintenance of operations	The Ground Segment shall provide components and interfaces for the maintenance of operational functions.
SR-1.1.1.56	Corrective maintenance	The Ground Segment shall ensure that minor defects can be easily and quickly identified and corrected.

SR-1.1.1.57	Adaptive maintenance	The Ground Segment shall ensure that future enhancements can be easily and quickly implemented.
SR-1.1.1.58	No down time during maintenance	The Ground Segment shall remain operational during all planned maintenance activities.
SR-1.1.1.59	Interface to management staff	The Ground Segment shall provide user interfaces to the staff responsible of its management.

SCALABILITY

SR-1.1.1.60	Load scalability	The Ground Segment shall be able to accommodate heavier or lighter loads (up to 100% variation) for all functionalities and interfaces supporting product generation and distribution.
SR-1.1.1.61	Heterogeneous scalability	The Ground Segment shall be able to scale up by using new generations of components, even from different vendors.

SECURITY

SR-1.1.1.62	Confidentiality	The Ground Segment shall assure confidentiality, and prevent the disclosure of information to unauthorized individuals or systems.
SR-1.1.1.63	Integrity	The Ground Segment shall provide data integrity, maintaining and assuring the accuracy and consistency of data over its entire life-cycle.
SR-1.1.1.64	Authenticity	The Ground Segment shall ensure authenticity, i.e. that its data, transactions, communications or documents are genuine.

2.3 INFORMATION VIEWPOINT

This section describes the system from the information viewpoint, which focuses on the semantics of the information and the information processing performed. It describes the information managed by the system and the structure and content type of the supporting data.

The Ground Segment shall support the space applications and their data model. Figure below shows a diagram describing the data model managed by the applications.

Green classes are describing data related to the Flight Operations (and thus, related to the Space Segment as well). A “satellite” flying a payload with several instruments, will receive tele-commands (“TC”) as part of a “schedule” or plan, and will send telemetry (“TM”) and space “events” characterizing its current “state”.

Blue classes are more related to the PDGS. The airborne “instruments” will generate “RAW” data acquisitions, which will produce data “products” and its corresponding “metadata” thanks to the support of a series of Auxiliary Data Files (“ADF”). These products would be then refined throughout a processing chain, from Level-0 (“L0”) to Level 2 or higher products (“L2+”).

Magenta classes represent those concepts related to the User Segment. An “EO user”, a representative of the data-products user community could request a series of products by means of an “order”.

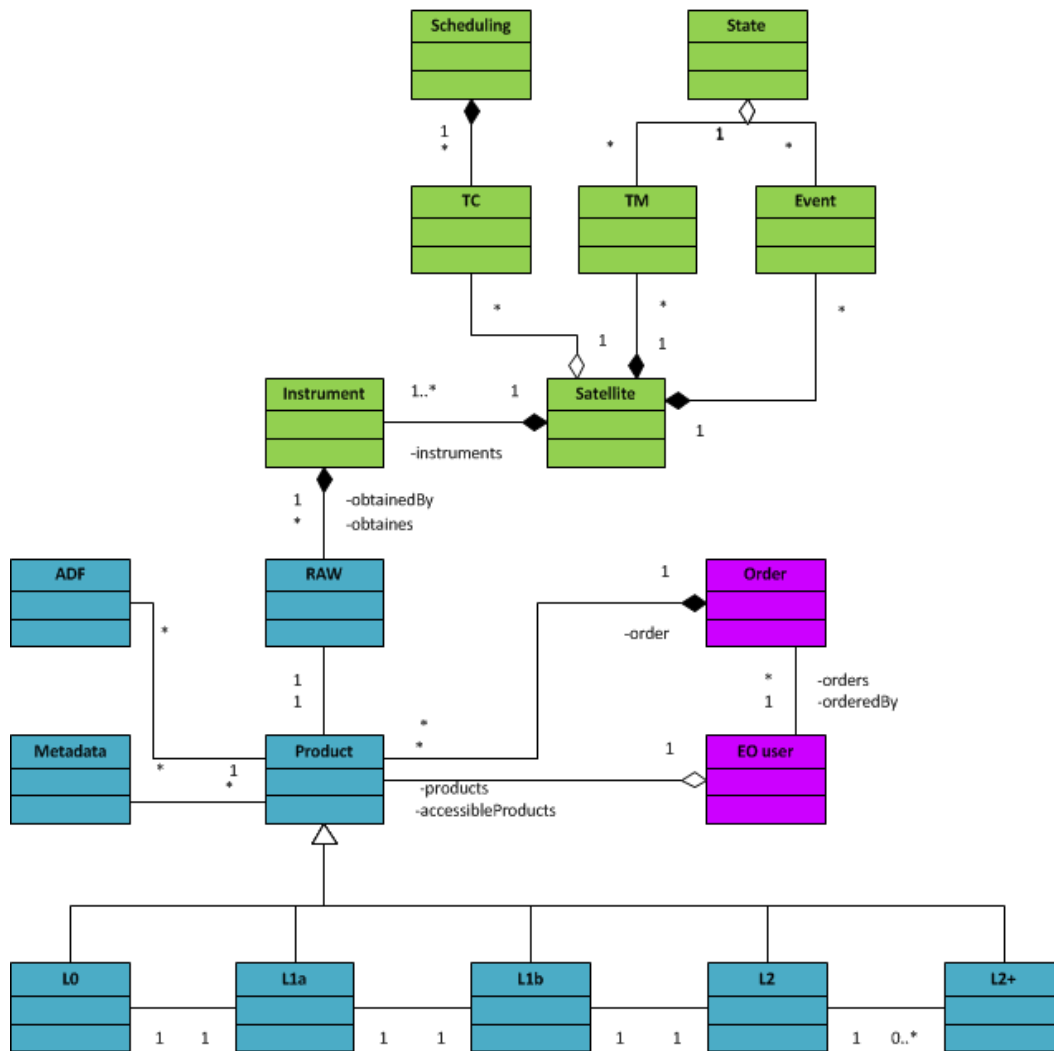


Illustration 2: Ground segment data model

The aforementioned data model is describing application data and the complete Enterprise Management (EM) shall support this via a middleware Service Enterprise Bus. This means that all those application data classes will be represented by:

- **Data objects (files)** – For instance “RAW” acquisitions and payload “products” are usually large chunks of binary or text data and will be considered as files, managed by a “repository service” located in the “service” layer, which would give access to databases and a distributed file system as represented in the “data” layer.
- **Query / results (messages)** - For instance “TC”, “TM” and satellite “Events” are small signals, maybe represented in a text, human-readable format, managed by a messaging middleware, which would give access to a series of services in the “services” layer.
- **Metadata / resources** – Support information describing data objects and listing the system resources registered in the “Service Registry” of the “service” layer and ready to be used by the applications.

2.4 COMPUTATIONAL VIEWPOINT

This section describes the computational viewpoint, which enables distribution through functional decomposition on the system into objects which interact at interfaces. It describes the functionality provided by the system and its functional decomposition.

The proposed Ground Segment and the Enterprise Management in particular are decomposed in a “layered architecture” to abstract unnecessary details from one context to another, separating ideas from specific instances of those ideas at work. Functional objects are defined by their meanings (semantics), while hiding away the details of how they work. In this case, the Ground Segment system has several abstraction layers whereby different meanings and amounts of detail are exposed to the users; low-level abstraction layers (“platform” and “infrastructure” layers) expose details of the computer hardware and operating systems where the programs runs, while high-level layers (“service”, “service integration” and “applications”) deal with the business logic of the system.

The selected architectural style providing a set of different levels of abstraction:

- simplifies the design considerably, as it hides the complexity details of other layers;
- enables different role players to effectively work at various levels of abstraction, avoiding the figure of “know-it-all” users with expertise in the whole system rather than specialists in their own areas;
- supports the portability of software and system artifacts, enabling the system to export these components to other systems;
- Promotes the substitution or upgrade of isolated layers without affecting the others.

The proposed Ground Segment layered architecture defines the following functional layers:

- **User interfaces layer** – provides single points of access for GS functions accessible to actors external to the system, i.e., controlling and monitoring EO satellites by means of ground stations, attending orders coming from EO users and replying with sets of products.
- **Application layer** – hosts and manages the PD, PDG and FO systems, maybe developed by third-parties, and providing the high-level functionality of a Ground Segment.
- **Service Integration layer** - exposes the services in the architecture in a consistent manner while enabling services to be implemented in a variety of technologies. By leveraging services available in the architecture, applications shall be quicker to develop and easier to maintain.
- **Services layer** - separates GS applications from re-usable services with clearly defined contracts that can be used by any application.
- **Data Integration layer** – gives a logically consistent way of interacting with data domains (i.e. enterprise, operational and product data), and a coherent strategy for sharing data across systems.
- **Data layer** - consolidates related data sets by data domain. Data shall not need to be physically located in the same database or file system, or managed by the same system, but that there

shall be a coherent set of rules for locating, a unified view of, and a standard way to access operational, product and enterprise data.

- **Platform layer** - provides all system software needed to manage the infrastructure and provide support to GS applications, services and data.
- **Infrastructure layer** - provides all GS hardware (servers, client devices, storage devices and network devices).
- **Security Services layer** - provides necessary services to apply access and control security to the infrastructure, to the platform, to data, to services, to applications and finally to user interfaces.

Following sections provide descriptions of each functionality grouped in layers. Descriptions of generally-used functions are mostly taken and adapted from the English Wikipedia, the rest are definitions provided by the author.

Illustration 2 provides a high-level overview of this functional decomposition in layers and the relationships between them.

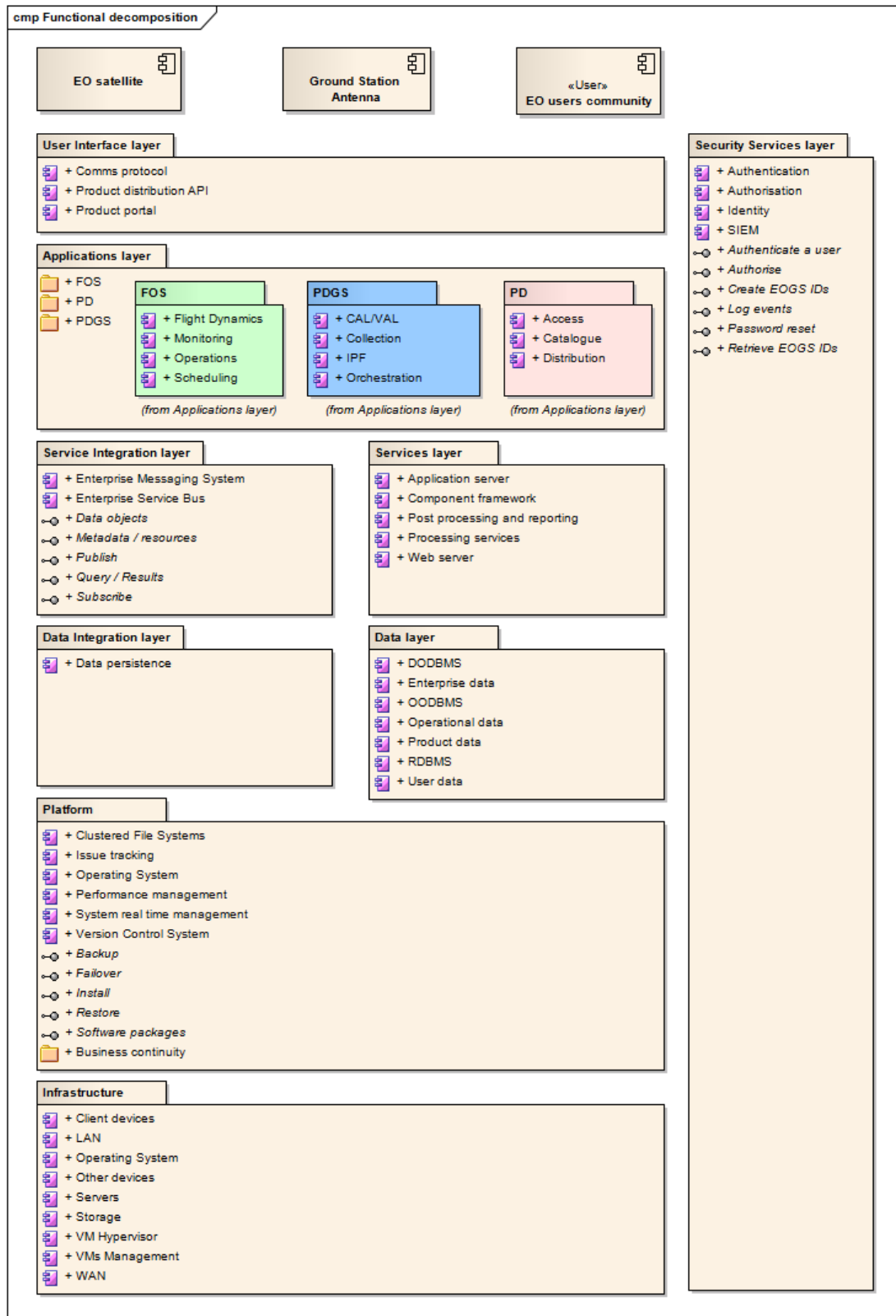


Illustration 3: Functional decomposition

2.4.1 USER INTERFACES LAYER

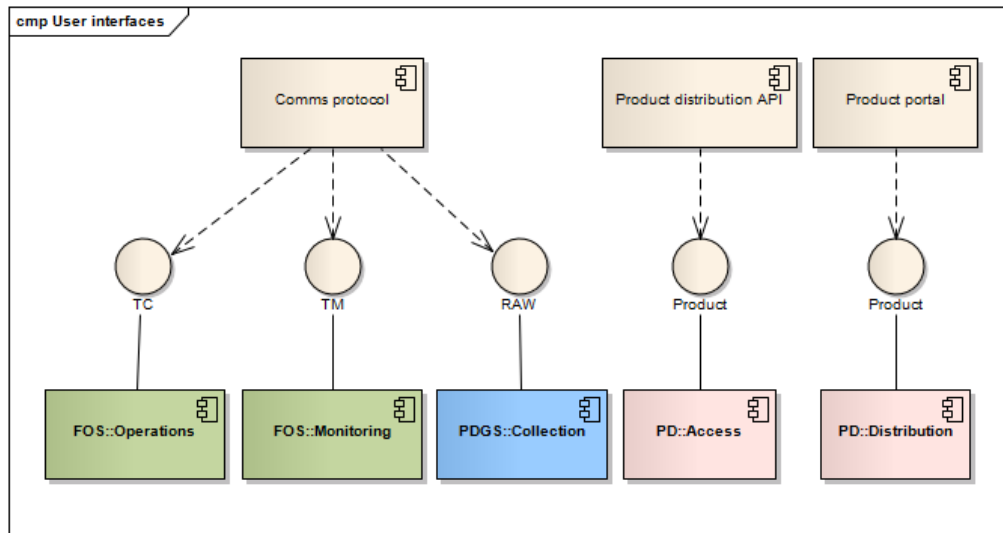


Illustration 4: User interfaces components

- Reception (acquisition) of RAW streams of data containing sensed parameters for the on-board instruments by using a, for instance, X-band antenna.

This function is the Ground Segment interface with the “Space Segment”.

PRODUCT DISTRIBUTION API

This function publishes an external Application Protocol Interface (API) following the Publish-Subscribe pattern to distribute products to subscribed users. This function is triggered by the generation of a product pertaining to a certain class that some user has expressed interest on it (“subscribed”). This function provides an interface with the external “EO user community”.

PRODUCT PORTAL/WEBSITE

This function provides an attractive means of consulting the available catalogue of data products and obtains a series of products, by placing an “order”. This function provides an interface with the external “EO user community”.

This diagram shows the minimal set of components in the “user interfaces layer”.

COMMUNICATION PROTOCOLS

This functionality provides a communication channel connecting a ground antenna and a satellite in orbit. This function supports of set of defined protocols and runs over a series of physical devices like modulators-demodulators (modems), ground antenna facilities and associated equipment.

Both the communications protocols and antenna specifications are outside of the scope of this project.

These protocols permit:

- Transmission of tele-commands (TC) to the satellite and reception of telemetry and events (TM) from the satellite using a, for instance, S-band ground antenna.

2.4.2 APPLICATIONS LAYER

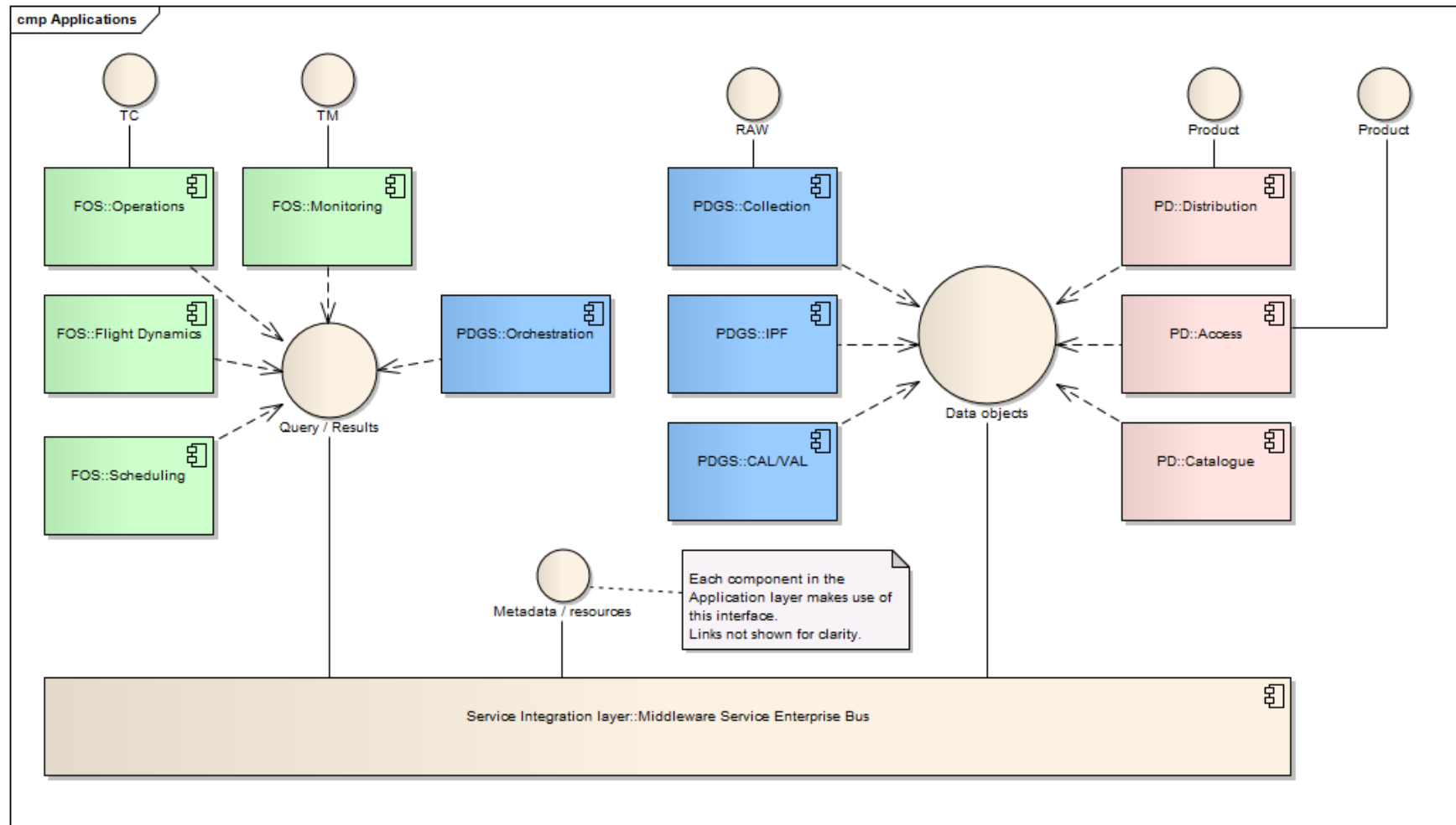


Illustration 5: Applications layer components

This diagram shows the minimal set of components for the Applications layer.

It is important to note that there are no point-to-point interactions between applications or facilities. Applications communicate with each other by a “publish/subscribe” messaging pattern using a common bus, acting as broker for all interactions in the form of messages and events. Applications do not unnecessarily replicate or move data files as all components can access the same distributed file systems and database services.

All components could make use of a third interface (besides query/results and data object files) that allows obtaining metadata (data about data files) and the list of available resources or services in the rest of the system.

The following functions correspond with the high-level functional requirements as specified for this mission.

FLIGHT OPERATIONS SEGMENT (FOS)

The Flight Operations Segment controls the EO satellite through all mission phases, including the following operations:

- Satellite operation planning (scheduling) based upon the observation plans and predicted orbit (flight dynamics);
- Command and control of the EO satellite (operations and monitoring).

FLIGHT DYNAMICS

Flight Dynamics provides functionality for orbit prediction, restitution and maintenance for nominal cases and for collision avoidance manoeuvres.

MONITORING

Monitoring receives telemetry measures and events with current satellite status, configuration and performance.

Published interfaces:

- **TM** – Telemetry defined in a well-known format.

OPERATIONS

The Operations system translates the mission plan into tele-commands ready to be sent to the EO satellite.

Published interfaces:

- **TC** – Tele-commands defined in a well-known format.

SCHEDULING

Scheduling prepares the satellite operations (platform and payload) plan.

PAYLOAD DATA GENERATION SEGMENT (PDGS)

This segment is responsible for the exploitation of the instrument data. The PDGS includes the facilities responsible for quality control (CAL/VAL), acquisition (Collection) and processing (Orchestration and IPF).

COLLECTION

This facility obtains and decrypts RAW data from the on-board instruments and ingests it into the internal catalogue

Published interfaces:

- **RAW** – Acquired, encrypted instrument data in engineering format.

INSTRUMENT PROCESSING FACILITY (IPF)

This facility processes and refines obtained data into products of increasingly added value.

ORCHESTRATION

Orchestration manages, controls and monitors all PDGS operations, including collection, processing and calibration/validation.

CALIBRATION AND VALIDATION (CAL/VAL)

This functionality is in responsible for calibration, validation, quality control and end-to-end system performance assessment.



THERE IS NO POINT-TO-POINT COMMUNICATIONS BETWEEN APPLICATIONS, ALL MESSAGES ARE COORDINATED BY A COMMON BUS.

MOREOVER, THERE IS NO UNNECESSARY MOVEMENT AND REPLICATION OF DATA BETWEEN SYSTEMS.

PRODUCT DISTRIBUTION SEGMENT (PDS)

The Product Distribution Segment sends requested data products to users by either a data push (dissemination) or data push (user initiated access). It also offers an indexed archive of data and associated meta-data.

ACCESS

This functionality provides an interface for users to access the catalogue and order archived or new data products (data pulling).

Published interfaces:

- **Product** – Data file and associated meta-data.

CATALOGUE

This functionality provides an indexed archive of data and associated meta-data.

DISTRIBUTION (DISSEMINATION)

This functionality sends requested data products to users by a data push. This case is more common of scientific institutions with open standing orders and continuous use of data.

Published interfaces:

- **Product** – Data file and associated meta-data.

2.4.3 SERVICE INTEGRATION LAYER

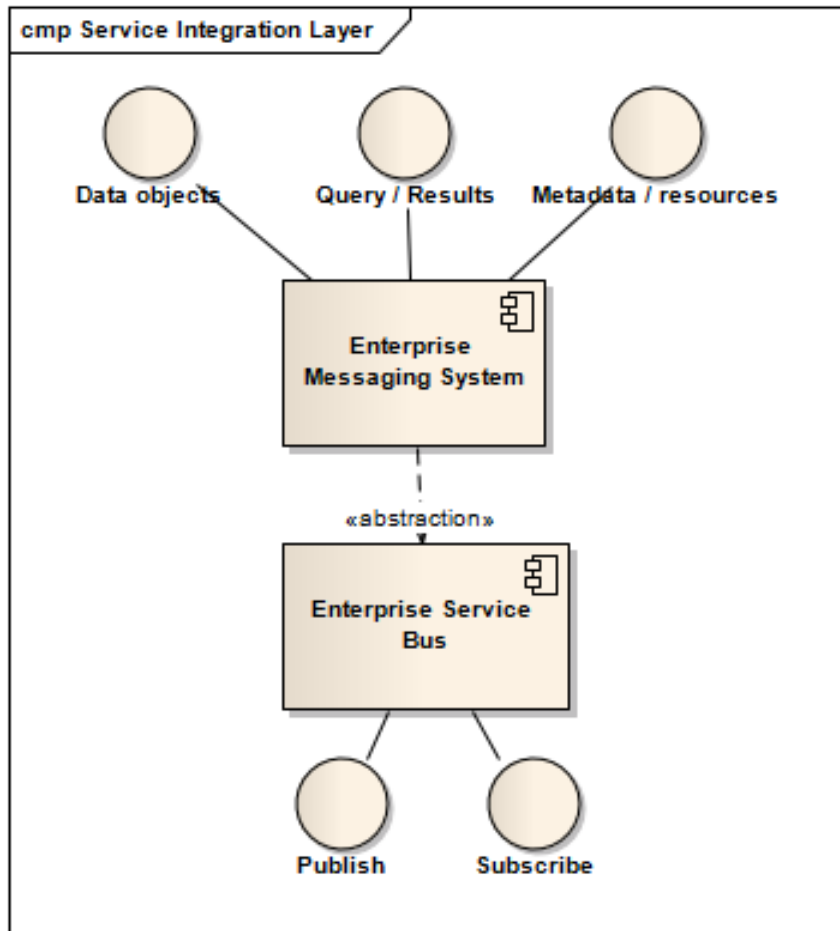


Illustration 6: Service integration layer components

This diagram shows the minimal set of components for the Service Integration layer.

ENTERPRISE SERVICE BUS

The architecture of this layer is based on an “Enterprise Service Bus”, that is a software architecture model used for designing and implementing communications between mutually interacting software applications in a service-oriented architecture (SOA). As software architectural model for distributed computing it is a specialty variant of the more general client server model and promotes agility and flexibility with regards to communication between applications. Its primary use is in enterprise application integration (EAI) of heterogeneous and complex landscapes.

In this architecture, source systems “publish” enterprise messages to a common bus, the Enterprise Service Bus; applications and rest of services “subscribe” to relevant messages and act on them.

This bus functionality acts as a broker between applications, provides near real-time, guaranteed, once-only delivery of messages, that can be later stored in a dedicated database for further analysis. This bus also provides a common environment in which to define rules.

The main advantages of this architecture is that the systems are integrated but not coupled; allows for near real-time integrations which reduced latency and that solves the exponential-escalation problem, as the integration effort expands linearly as the number of systems



HIGH-LEVEL, SPACE-RELATED BUSINESS RULES WOULD BE DEFINED AND MANAGED BY THE ENTERPRISE SERVICE BUS, THAT WOULD CALL ATOMIC OPERATIONS PROVIDED BY APPLICATIONS AND SERVICES

increases.

Public interfaces:

- **Publish message** – An application or service generates a class of message, without knowledge of what, if any, subscribers there may be.
- **Subscribe to message** – An application or service expresses interest in one or more classes of messages, without knowledge of what, if any, publishers there are.

In EOGS, this ESG provides an abstraction layer on top of an implementation of an enterprise messaging system, that is the one responsible of breaking up those basic functions into their constituent and atomic parts, according to the enterprise business logic.

ENTERPRISE MESSAGING SYSTEM

An enterprise messaging system (EMS) is a function that allows organizations to send semantically meaningful messages between computer systems. EMS systems promote loosely coupled architectures that allow changes in the formats of messages to have minimum impact on message subscribers.

Although similar in concept to an enterprise service bus (ESB), an EMS places emphasis on design of messaging protocols (for instance, using DDS, MSMQ or AMQP), not the implementation of the services using a specific technology such as web services, DDS APIs for C/C++ and Java, .Net Framework or Java Message Service (JMS).

In EOGS, the messages are related to:

- Data Objects (files)
- Queries and their results
- Retrieval of metadata and resources

Note: The reasoning behind the design of this extra layers of abstraction for messages and communications between applications, even though those functionalities are already provided by the common component framework in the “Services layer”, is to be able to contact with other systems not following the same approach, as it would be the case with satellite links, remote processing facilities or legacy applications.



IT WILL BE POSSIBLE TO KNOW THE PRECISE STATUS OF THE WHOLE GROUND SEGMENT AT ANY PRESENT OR PAST TIME, FACILITATING DATA PERSISTENCE AND ANALYSIS

2.4.4 SERVICES LAYER

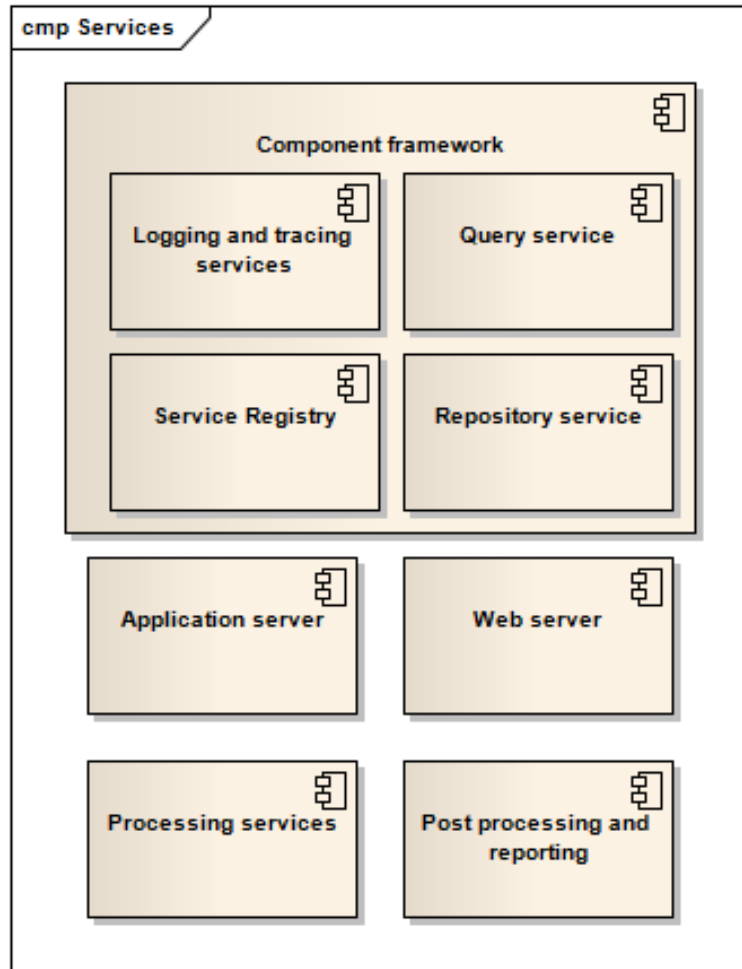


Illustration 7: Services layer components

This diagram shows a minimal set of components for the Services layer.

The architecture of this layer is a “component-based system” as all system functions are placed into separate components (i.e. software packages, web services, web resources or modules) so that all of the data and functions inside each component are semantically related.

A component model is a definition of standards for component implementation, documentation and deployment.

COMPONENT FRAMEWORK

A component framework describes a modular system and a system platform that implements a complete and dynamic component model. Applications or components, coming in the form of bundles for deployment, can be remotely installed, started, stopped, updated, and uninstalled without requiring a reboot.

In EOGS, the selected component framework (OSGi) provides a series of standard services and the possibility to use extended services that will be managed by the Service Registry.

LOGGING AND TRACING SERVICES

This function provides the logging of information, warnings, debugging messages or errors and is handled through the OSGi Log Service. It receives log entries and then dispatches these entries to other components that subscribed to this information.

Public interfaces:

- Log events

QUERY SERVICES

The query service manages routing of queries in order to discover and locate the rest of services providing data files, messages and meta-data which contain information to satisfy user queries. Once the service has returned the information that satisfies the query, the information is aggregated and returned to the query service. At that point, the query service can perform processing such as packaging, translations to other formats, and other types of advanced processing.

Published interfaces:

- Route query

SERVICE REGISTRY

The service registry provides an interface to search for functional services that perform a needed action specified by a user. Service registries manage descriptions of service interfaces (called service descriptions), including their respective locations, methods, and method parameters. Service descriptions are important because they describe software methods, software systems, and Web resources using metadata. Because of this, they can be queried to retrieve a service endpoint (essentially a pointer to the service's location), and metadata describing how to invoke the particular service. This helps to facilitate the use and consumption of services dynamically via software rather than explicit invocations and requests.

Published interfaces:

- Locate service

REPOSITORY SERVICES

A repository service is responsible for management of an underlying data store object or the physical data store. The repository service object differs from a data store object by several properties that are typically considered non-functional. These properties include scalability, dependability, uniformity, and other quality attributes. In this context, repository service objects provide the same get and put methods that the data store object provides. However, whereas a data store object may not scale across many underlying physical data stores, may not be dependable 24x7, and may not provide a uniform software interface, a repository service object is responsible for delivering necessary quality of service in each of these non-functional properties.

Published interfaces:

- Get a data store object



**OSGI IS THE SELECTED COMPONENT FRAMEWORK
FOR DEPLOYING AND SUPPORTING SPACE-
RELATED APPLICATIONS**

- Put a data store object

APPLICATION SERVER

An application server is, in this case, the server portion of a specific implementation instance of a software framework that provides a generalized approach to creating an application-server implementation, regardless to what the application functions are. The server's function is dedicated to the efficient execution of procedures (programs, routines, scripts) for supporting its applied applications.

PROCESSING SERVICES

In this case, the processing service provides a framework for large-scale processing of data-sets on clusters of commodity hardware. Processing is made following the Map Reduce paradigm, which is a programming model suitable for this processing of parallel, distributed algorithms on a cluster.

A Map Reduce program is composed of a Map procedure that performs sorting and filtering (such as dividing image products in cells and bands and queuing them for a cloud masking operation) and a Reduce procedure that performs a summary operation (such as mosaicking the resulting cloud-free images). The "Map Reduce System" (also called "infrastructure" or "framework") orchestrates by marshalling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance.

Published interfaces:


- Run process
- Stop process

POST-PROCESSING AND REPORTING SERVICES

This service provides reporting and business intelligence capabilities for applications. Business intelligence capabilities range from operational or enterprise reporting to multi-dimensional online analytical processing (OLAP).

WEB SERVER

This service helps to deliver web content accessible through HTTP from the internal networks.



**HADOOP WILL BE CHOSEN AS FRAMEWORK FOR A
LARGE-SCALE PROCESSING FOR BOTH PARALLEL
AND BATCHED EXECUTION OF JOBS**

2.4.5 DATA INTEGRATION LAYER

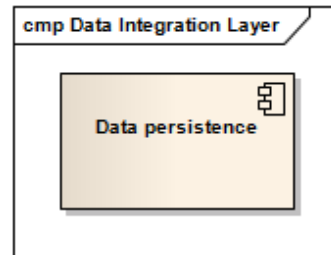


Illustration 8: Data integration layer components

This diagram shows a minimal set of components for the data integration layer.

DATA PERSISTENCE

This function provides a mapping framework between language objects and databases (either relational, object-oriented databases or even files). One of its features is the transparency of the persistence services to the domain model.

Published interfaces:

- Put data
- Get data

2.4.6 DATA LAYER

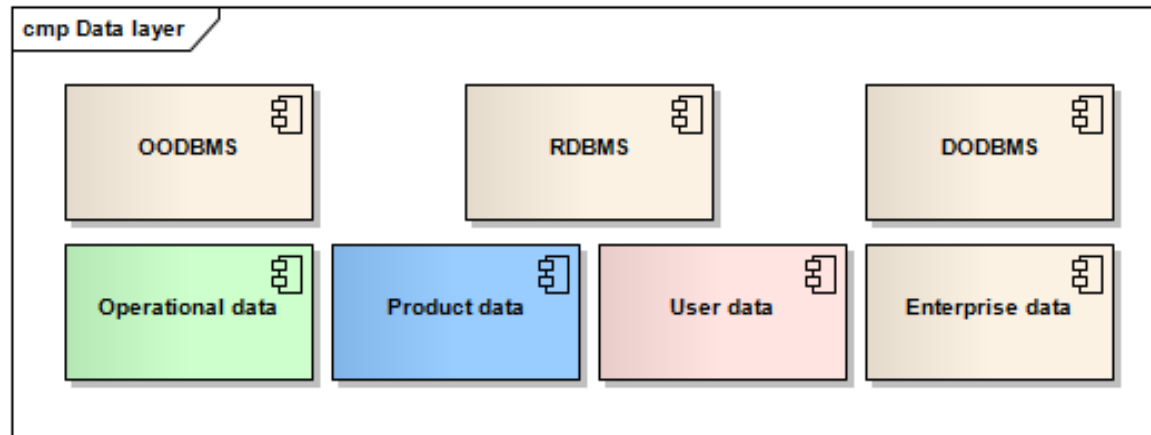


Illustration 9: Data layer components

This diagram shows the minimal set of components of the data layer.

RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)

This function is a database management system (DBMS) that is based on the relational model. The relational model for database management is a database model based on first-order predicate logic. In the relational model of a database, all data is represented in terms of tuples, grouped into relations.

In EOGS, this function would be used for storing structured information with complex querying requirements, which would be solved by use of SQL language.

OBJECT ORIENTED DATABASE (OODB) MANAGEMENT SYSTEM

This function provides a database management system in which information is represented in the form of objects as used in object-oriented programming. Object databases are different from relational databases which are table-oriented.

In EOGS, this OODB would be used for complex data structures, like graphs or key-value pairs, not able or worth to be represented in tables.

FILE (DOCUMENT)-ORIENTED DATABASE MANAGEMENT SYSTEM

A document-oriented database is a computer program designed for storing, retrieving, and managing document-oriented information, also known as semi-structured data. Document-oriented databases are one of the main categories of NoSQL databases and the popularity of the term "document-oriented database" (or "document store") has grown with the use of the term NoSQL itself. In contrast to relational databases and their notions of "Relations" (or "Tables"), these systems are designed around an abstract notion of a "Document".

In EOGS this functionality would be used to index big data files, like EO products.

ENTERPRISE DATA

Data store containing information generated by the Enterprise Segment and to be used for its operations.

OPERATIONAL DATA

Data store containing information related to the Flight Operations Segment, including TMs, TCs, events, orbit, plans, etc.

PRODUCT DATA

Data store containing information related to the Payload Data Generation Segment, including all products (from RAW to L2+), calibration and validation reports, etc.

USER DATA

Data store containing information related to the Product Distribution Segment, including users, orders, etc. It includes a roles database for describing user profiles and authorisation rights.

2.4.7 PLATFORM LAYER

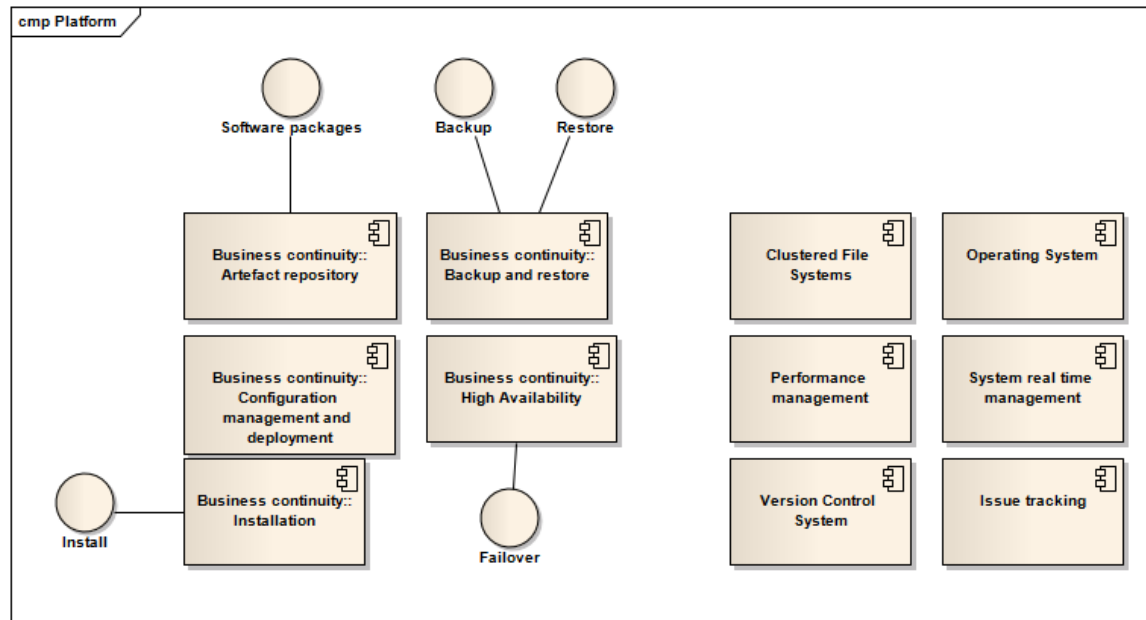


Illustration 10: Platform layer components

(including applications) are recovered by means of a quick Deployment of software packages (installation packages) hosted and on an Artefact Repository.

3. Contingency: plans designed to cope effectively with whatever major incidents and disasters occur. These plans are out of the scope of this project, but a good Ground Segment implementation should always have them in place.

ARTEFACT REPOSITORY

An Artefact (or Software) Repository is a storage location from which software packages may be retrieved and installed on a computer system. This function is used on EOGS to host and serve the latest valid versions of the software components present in the system. Software components are distributed and installed in the form of installation packages. A Versioning Control System is used in conjunction with this Repository to keep track of the different baselines and versions history of each of the identified Configuration Items, including software packages.

This diagram shows a minimal set of components of the platform layer.

BUSINESS CONTINUITY

Business continuity groups a set of functions to ensure that the EOGS system will either continue to operate despite serious incidents or disasters that might otherwise have interrupted them, or will be recovered to an operational state within a reasonably short period.

There are three key elements:

1. Resiliency: the system is created in such a way that it is unaffected by most disruptions, for example through the use of redundancy and spare capacity. This element is covered on EOGS by a High Availability function and data redundancy of the distributed file systems.
2. Recovery: methods to recover or restore critical and less critical business functions that fail for some reason. In EOGS, data is recovered from historical backups and archives through a B&R function, whereas software components

Public interfaces:

- **CRUD** functions, i.e., Create, Read, Update and Delete **software packages**

BACKUP AND RESTORE

In information technology, a backup, or the process of backing up, refers to the copying and archiving of computer data so it may be used to restore the original after a data loss event.

Backups have two distinct purposes:

- The primary purpose is to recover data after its loss, be it by data deletion or corruption.
- The secondary purpose of backups is to recover data from an earlier time, according to a user-defined data retention policy, typically configured within a backup application for how long copies of data are required.

In EOGS, only data would be protected by means of a backup/restore function, whereas other software components will be reconstructed by means of a Configuration Management/Deployment function.

Public interfaces:

- **Backup data** – Copy data of its current status on a dedicated, isolated and secured storage device.
- **Restore previous data** – Substitute current data (that might be corrupted, lost or simple wrong) by a previous version of it, with a maximum antiquity defined on a Retention plan.

SOFTWARE CONFIGURATION MANAGEMENT AND DEPLOYMENT

In software engineering, software configuration management (SCM) is the task of tracking and controlling changes in the software, part of the larger cross-discipline field of configuration management. SCM practices include revision control and the establishment of baselines. If something goes wrong, SCM can determine what was changed and who changed it. If a configuration is working well, SCM can determine how to replicate it across many hosts.

In EOGS, "software configuration" is understood to cover changes typically made by a system administrator; management of source code undergoing software development is considered a separate function covered by a "Version Control System".

DATA WILL BE PROTECTED BY A BACKUP AND RESTORE MECHANISM BASED ON FILE SYSTEM SNAPSHOTS, WHEREAS APPLICATIONS WILL BE RECOVERED VIA A FAST DEPLOYMENT PROCESS

The configuration, once is considered to be “working well”, it would be deployed to other nodes through a Deployment function.

HIGH AVAILABILITY

High Availability is a characteristic of a system, defined by the ratio between up time (calculated as total time minus down time) and the total time.

To ensure meeting the downtime requirements, the EOGS system would use a High Availability Cluster, which is a group of computers providing a list of services or software applications with a minimum down-time. This cluster provides node redundancy, a quick fault-detection system (usually by thanks to a “heart-beat” method) and a reliable fail-over mechanism to another resource in case a failure is detected on an active one.

Public interfaces:

- **Failover** – Ensures a service or application is back online after a failure with the same status as it had before.

INSTALLATION (SOFTWARE PACKAGES)

Installation of a software is the act of making the software ready for execution on a computer system. Because of the various processes needed to do so, EOGS would make use of software packages for their installations.

A software package is a software that has been built from source with one of the available package management systems (PMS). The package is typically provided as compiled code, with additional meta-information such as a package description, package version, or "dependencies". The package management system can evaluate this meta-information to allow package searches; to perform automatic upgrades to a newer version; to check that all dependencies of a package are fulfilled and/or to fulfil them automatically by installing missing packages.

EOGS would deploy and install software packages kept on a software repository by means of a configuration management system.

Public interfaces:

- **Install software** – Makes a software available for execution on a certain computer system.

CLUSTERED FILE SYSTEM

A clustered file system is a file system which is shared by being simultaneously mounted on multiple servers.

Clustered file systems can provide features like location-independent addressing and redundancy which improve reliability or reduce the complexity of the other parts of the cluster.

Two kinds of clustered file systems would be used for EOGS:

- **Network attached Storage (NAS)** - Network attached storage (NAS) provides both storage and a file system, like a shared disk file system on top of a storage area network (SAN). NAS typically uses file-based protocols (as opposed to block-based protocols a SAN would use).

In EOGS, it would be used for the processing services, as processing nodes would use their local disk drives and shared with the rest of the processing cluster via file-based protocols.

- **Shared-disk / Storage Area Network (SAN)** - A shared-disk file system uses a storage-area network (SAN) to provide direct disk access from multiple computers at the block level. Access control and translation from file-level operations that applications use to block-level operations used by the SAN must take place on the client node. The most common type of clustered file systems is shared-disk file system, which—by adding mechanisms for concurrency control—provides a consistent and serializable view of the file system, avoiding corruption and unintended data loss even when multiple clients try to access the same files at the same time.

In EOGS, it would be used for storing the data of the rest of operational and enterprise services, ready to be shared across all the segments.

ISSUE TRACKING

An Issue tracking system is a computer software package that manages and maintains lists of issues (problems, bugs, tickets, requests), as needed by an organization. An issue tracking system often also contains a knowledge base containing information on each customer, resolutions to common problems, and other such data.

In EOGS, it would be used to keep track of software and operational problems and requests from the user segment.

OPERATING SYSTEM (OS)

An operating system (OS) is software that manages computer hardware resources and provides common services for computer programs. In this case it refers to the client Operating System deployed on guest Virtual Machines.

PERFORMANCE MANAGEMENT

This function is a computer system monitoring, network monitoring and infrastructure monitoring software application. It offers monitoring and alerting services for servers, switches, applications, and services. It alerts the users when things go wrong and alerts them a second time when the problem has been resolved.

SYSTEMS REAL TIME MANAGEMENT


This function supplies means for managing and monitoring applications, system objects, devices (e.g. printers) and service oriented networks.

This function is complementary to the one provided by the Performance Management as it is more oriented to software applications, where the other is for hardware devices.

VERSION (OR REVISION) CONTROL SYSTEM

The Revision Control System (RCS) is a software implementation of revision control that automates the storing, retrieval, logging, identification, and merging of revisions. RCS is useful for text that is revised frequently, for example configuration files, documentation, procedural graphics, papers, and form letters. RCS is also capable of handling binary files, though with reduced efficiency.

In EOGS this function would be used to control text file baselines.



BOTH THE PERFORMANCE MANAGEMENT, THE SYSTEMS REAL TIME MANAGEMENT AND THE SECURITY INFORMATION AND EVENT MANAGEMENT SYSTEMS (DESCRIBED LATER) WOULD BE CORE PROVIDERS OF INFORMATION FOR HIGH-LEVEL MONITORING APPLICATIONS

2.4.8 INFRASTRUCTURE LAYER

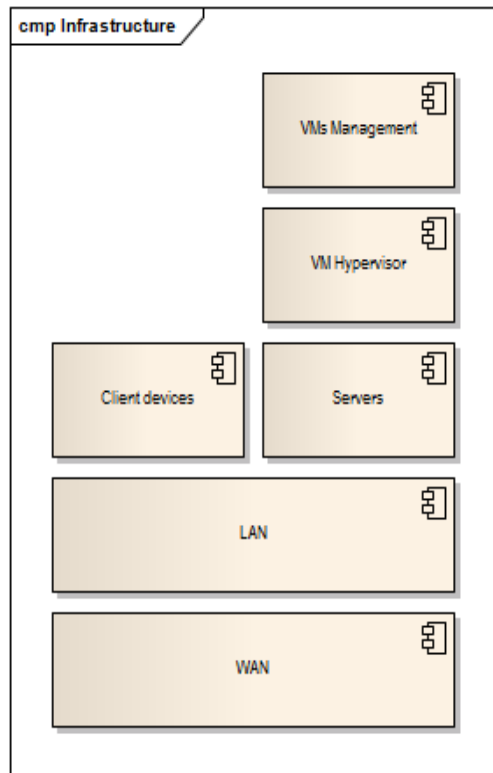


Illustration 11: Infrastructure layer components

This diagram shows a minimal set of components of the infrastructure layer.

VM MANAGEMENT

This function manages platform virtualization. Computer hardware virtualization is the virtualization of computers or operating systems. It hides the physical characteristics of a computing platform from users, instead showing another abstract computing platform. A VM manager allows users to create, edit, start and stop VMS and view and control each VM’s console, among many other functions.

VM HYPERVISOR

A hypervisor or virtual machine monitor (VMM) is a piece of computer software, firmware or hardware that creates and runs virtual machines.

A computer on which a hypervisor is running one or more virtual machines is defined as a host machine. Each virtual machine is called a guest machine. The hypervisor presents the guest operating systems with a virtual operating platform and manages the execution of the guest operating systems. Multiple instances of a variety of operating systems may share the virtualized hardware resources.

OPERATING SYSTEM

An operating system (OS) is software that manages computer hardware resources and provides common services for computer programs. In this case it refers to the host Operating System that would run the virtual machine hypervisor. There is also the option that the technology chosen for the VM hypervisors is one of the “Type 1” (or native, bare metal) where it runs directly on the host’s hardware to control the hardware and to manage guest

operating systems. A guest operating-system of that type would run on another level above the hypervisor.

Besides the software components of this layer, there is also a series of hardware devices, including servers, client devices, storage devices, LAN devices (routers, hubs, etc.), WAN devices (routers, firewalls, etc.) and other devices like NTP servers, KVM switches (Keyboard, Video and Mouse), cooling system, Uninterrupted Power Systems (UPS), temperature sensors, etc. Their definition is out of the scope of this project.



PROCESSING SERVICES WILL NOT BE VIRTUALIZED, AS THEY ALREADY OFFER THEIR OWN REDUNDANCY, AVAILABILITY AND SCALABILITY FUNCTIONS

2.4.9 SECURITY SERVICES LAYER

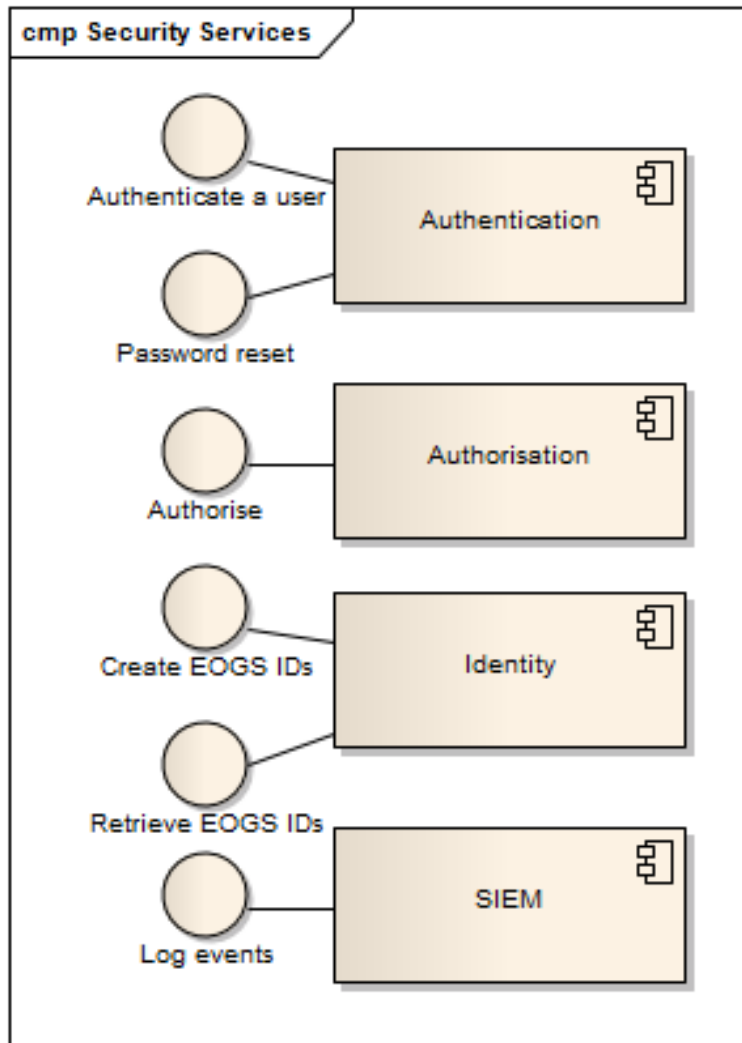


Illustration 12: Security services layer components

This diagram shows the minimal set of components of the security services layer.

AUTHENTICATION

Authentication is the process of identifying a user of a computer system. In security systems, authentication is distinct from authorisation, which is the process of giving individuals access to system objects based on their identity. Authentication merely ensures that the individual is who he or she claims to be, but says nothing about the access rights of the individual.

Public interfaces:

- **Authenticate a User** – Allows a component to authenticate the user (i.e., assert that they own the identity supplied)
- **Password Reset** – Allows a user to reset their password when locked out and attempting to access a component.

AUTHORISATION

Authorisation is the function of specifying access rights to resources related to information security and computer security in general and to access control in particular.

The formal way to grant authorisation to resources is defining an “access policy” to users grouped in “roles”, stored in a “Roles Database” of the “User Data” store.

Public interfaces:

- **Authorise a user-resource interaction** - The Authorisation component and the “Roles Database” provides a consistent way to store and maintain access rules for other components (i.e., applications and services). Components with an interface to the Authorisation component interpret the access rules from the “Roles Database” and enforce them.

IDENTITY

The function of identification is to map a known quantity to an unknown entity so as to make it known. The known quantity is called the identifier (or ID) and the unknown entity is what needs identification. A basic requirement for identification is that the ID be unique. IDs may be scoped, that is, they are unique only within a particular scope. IDs may also be built out of a collection of quantities such that they are unique on the collective.

Public interfaces:

- **Create EOGS IDs** – The EOGS ID is a 9 digit number used to uniquely identify any member of their user community, operators and external actors interacting with the system.
- **Retrieve EOGS IDs** – The EOGS ID is retrieved by supplying a person’s first and last name or only the name for a physical entity.

SECURITY INFORMATION AND EVENTS MANAGEMENT (SIEM)

SIEM functionality provides real-time analysis of security alerts generated by network hardware and software components. SIEM are also used to log security data and generate reports for compliance purposes.

The term SIEM describes the product capabilities of gathering, analysing and presenting information from network and security devices; identity and access management applications; vulnerability management and policy compliance tools; operating system, database and application logs; and external threat data. A key focus is to monitor and help manage user and service privileges, directory services and other system configuration changes; as well as providing log auditing and review and incident response.

Public interfaces:

- **Log events** - Log management aggregates events data from many sources, including network, security, servers, databases, applications, providing the ability to consolidate monitored data to help avoid missing crucial events.

2.5 ENGINEERING VIEWPOINT

This section provides the engineering viewpoint, which focuses on the mechanisms and functions required to support distributed interactions between objects in the system. It describes the distribution of processing performed by the system to manage the information and provide the functionality.

2.5.1 Data processing

In this case we are going to limit ourselves in showing the first steps of the “Data processing” scenario. The “business rules” for this scenario declare that when satellite data is collected, it would be later processed, validated and finally catalogued. The detailed steps are:

1. The “EO satellite” sends an instrument data signal through a “Downlink”.
2. The “Ground Station Antenna” receives that data signal.
3. The ground station decodes the signal as defined by the “Communication Protocol”.
4. The “Collection” function “acquires” and “decrypts” that “RAW” data. Then it sends a “Data Object” message to store the file.

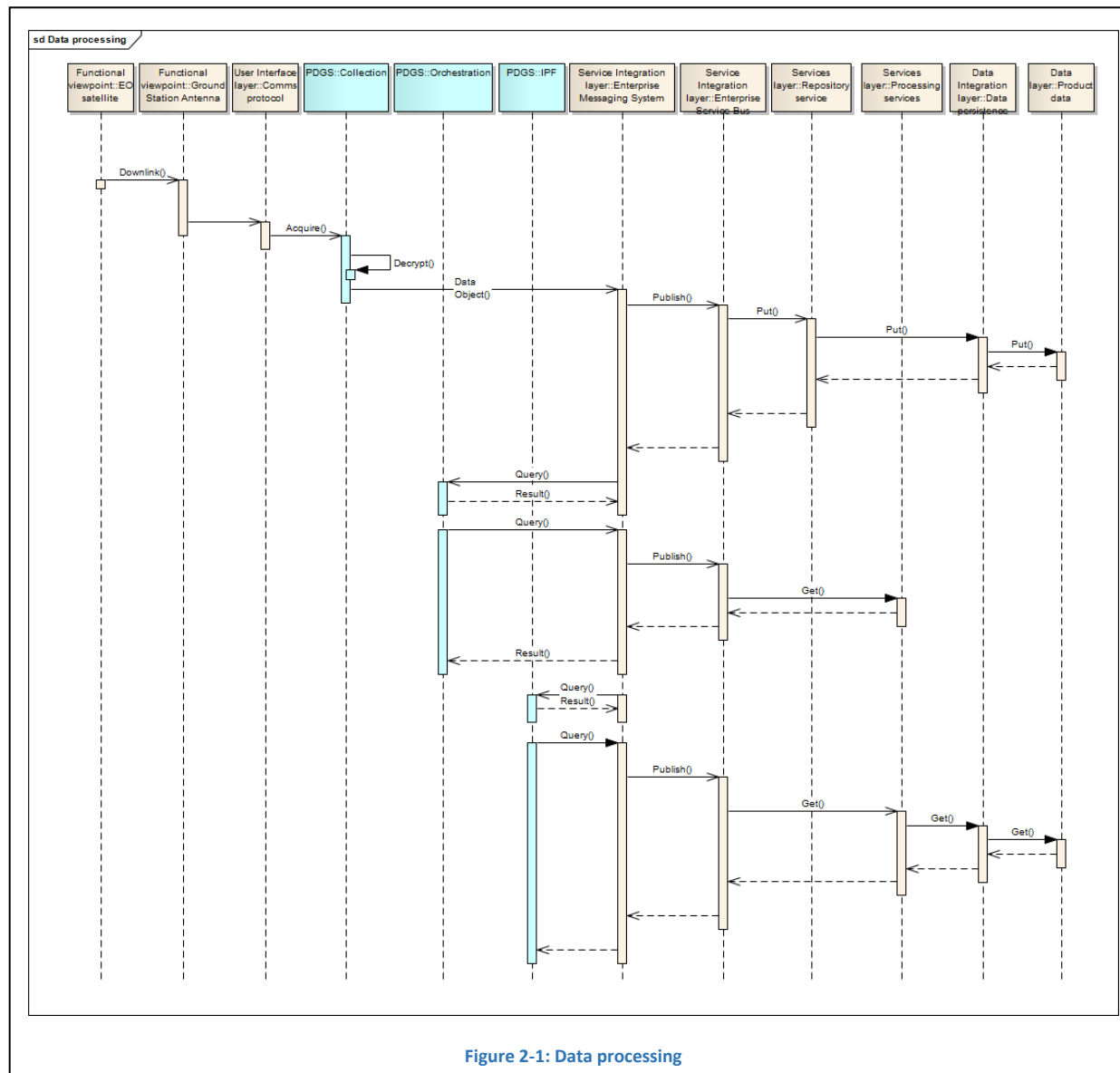


Figure 2-1: Data processing

5. The message is received by the “Enterprise Messaging System”, which would later “Publish” it to its subscribers.
6. One of its subscribers is the “Repository Service”, that would ask the data to be “put” into the “data persistence system”.
7. The “data persistence system” would later interact with the data store to actually insert or “put” that data file into its databases. Not shown: the File-oriented database system is the one in charge of finally storing it.
8. After the operation is declared finished, and as stated in the business rules, the “Enterprise Messaging Bus” would “query” the “Orchestrator” to start the processing of that data file present in the data storage. Thus, it sends a “Query” petition to the EMS, would then access the “Processing services” that would finally prepare the processing environment.
9. Next business rule says that the “Instrument Processing Facility (IPF)” should send to the “Processing services” a petition to execute the data with provided “algorithms”.
10. Not shown in the diagram: once the processing is finished the system would perform a validation process and, if successful, insert it in the general catalogue, now available to external users.

This scenario is used to clearly remark that there is no data movement between facilities, no unnecessary copies, and that every bit of data is stored in a common space.

It is worth noting that there is no point-to-point communications between the high-level applications and facilities, which all communications are made using asynchronous messages and a communication bus and end up in other applications or inner common services. Those common services will interact within themselves to provide the results of the queries and messages.

2.5.2 Satellite control

Another operational scenario would be controlling the satellite, in which these business rules are defined.

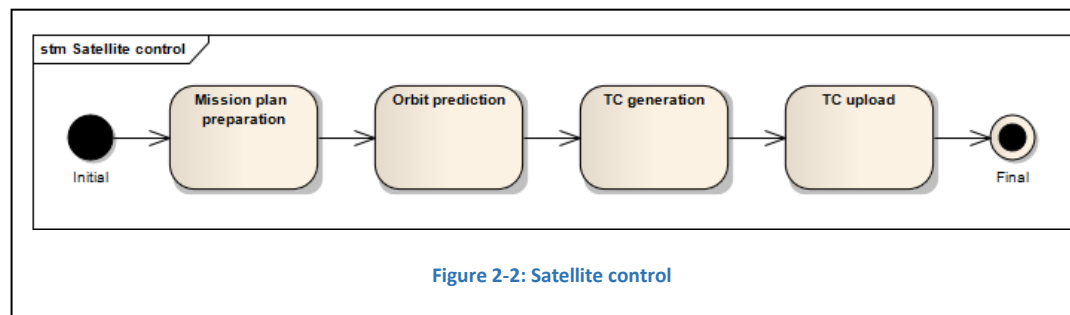


Figure 2-2: Satellite control

2.6 TECHNOLOGY VIEWPOINT

This section includes the technology viewpoint, which focuses on the choice of technology of the system. It describes the technologies chosen to provide the processing, functionality and presentation of information.

2.6.1 SOFTWARE

The following table includes the list of software components and technologies chosen to implement the needed functionalities. All descriptions are extracted from the English Wikipedia, unless stated differently.

LAYER	FUNCTIONALITY	CHOICE	DESCRIPTION
User interfaces layer	Web Site	Atlassian Confluence	Confluence is team collaboration software. Written in Java and mainly used in corporate environments. Confluence is sold as either on-premises software or as a hosted solution.
	User Interface		
	Thick Clients	Eclipse RCP	Eclipse provides the Rich Client Platform (RCP) for developing general purpose applications. A rich client platform (RCP) is a programmer tool that makes it easier to integrate independent software components, where most of the data processing occurs on the client side.
	Thin Clients	Eclipse RAP	Remote Application Platform (RAP, formerly Rich Ajax Platform) Project is an open-source software project under the Eclipse Technology Project which aims to enable software developers to build Ajax-enabled rich Internet applications by using the Eclipse development model, plugins and a Java-only application programming interface (API). It can be considered a counterpart for web development to the Rich Client Platform (RCP).
Application layer		N/A	

LAYER	FUNCTIONALITY	CHOICE	DESCRIPTION
Services integration layer	Service Integration Platform ESBs	Apache ServiceMix	<p>Apache ServiceMix is an enterprise-class open-source distributed enterprise service bus (ESB) based on the service-oriented architecture (SOA) model.</p> <p>ServiceMix 4 fully supports the OSGi framework.</p> <p>ServiceMix is often used with Apache ActiveMQ in SOA infrastructure projects.</p>
	Enterprise Messaging System	JMS and Apache ActiveMQ	<p>The Java Message Service (JMS) API is a Java Message Oriented Middleware (MOM) API for sending messages between two or more clients. It allows the communication between different components of a distributed application to be loosely coupled, reliable, and asynchronous.</p> <p>Apache ActiveMQ is an open source message broker written in Java together with a full Java Message Service (JMS) client. It provides "Enterprise Features" which in this case means fostering the communication from more than one client or server. Supported clients include Java via JMS 1.1 as well as several other "cross language" clients. The communication is managed with features such as computer clustering and ability to use any database as a JMS persistence provider besides virtual memory, cache, and journal persistency.</p>
Services layer	Component Framework	OSGi	<p>The OSGi (Open Service Gateway initiative) specification describes a module system and service platform for the Java programming language that implements a complete and dynamic component model, something that does not exist in standalone Java/VM environments. Applications or components, coming in the form of bundles for deployment, can be remotely installed, started, stopped, updated, and uninstalled without requiring a reboot; management of Java packages/classes is specified in great detail. Application life cycle management is implemented via APIs that allow for remote downloading of management policies. The service registry allows bundles to detect the addition of new services, or the removal of services, and adapt accordingly.</p>
	Application server	Eclipse Virgo	<p>Virgo is an open source, OSGi-based, Java application server originally developed as SpringSource dm</p>

LAYER	FUNCTIONALITY	CHOICE	DESCRIPTION
			<p>Server by SpringSource and currently maintained by the Eclipse Foundation. Virgo supports the deployment of OSGi bundles and unmodified Java web applications as well as OSGi-influenced Shared Libraries WARs and Shared Services WARs.</p> <p>Virgo is based on the Equinox OSGi implementation, part of the Eclipse project; the Apache Tomcat servlet container; The Spring Framework; and Spring Dynamic Modules for OSGi Service Platforms.</p>
	Processing services	Apache Hadoop	Apache Hadoop is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware. Hadoop is an Apache top-level project being built and used by a global community of contributors and users.
	Logging and Tracing services	OSGi Log Service	Part of the OSGi system services. The logging of information, warnings, debug information or errors is handled through the Log Service. It receives log entries and then dispatches these entries to other bundles that subscribed to this information.
	Post Processing and Reporting	Eclipse BIRT	The Business Intelligence and Reporting Tools (BIRT) Project is an open source software project that provides reporting and business intelligence capabilities for rich client and web applications, especially those based on Java and Java EE.
Data integration layer	Data Persistence	Java Data Objects	Java Data Objects (JDO) is a specification of Java object persistence. One of its features is a transparency of the persistence services to the domain model. JDO persistent objects are ordinary Java programming language classes (POJOs); there is no requirement for them to implement certain interfaces or extend from special classes.
Data layer	Data Archiving		
	RDBS	Postgresql	PostgreSQL, often simply "Postgres", is an object-relational database management system (ORDBMS) with an emphasis on extensibility and standards-compliance. As a database server, its primary function is to

LAYER	FUNCTIONALITY	CHOICE	DESCRIPTION
			store data, securely and supporting best practices, and retrieve it later, as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet). It can handle workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users. Recent versions also provide replication of the database itself for security and scalability.
	noSQL	Apache Cassandra	Apache Cassandra is an open source distributed database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. Cassandra offers robust support for clusters spanning multiple data-centres, with asynchronous master-less replication allowing low latency operations for all clients.
	Product file formats	HDF5	Hierarchical Data Format (HDF, HDF4, or HDF5) is a set of file formats and libraries designed to store and organize large amounts of numerical data. Originally developed at the National Centre for Supercomputing Applications, it is supported by the non-profit HDF Group, whose mission is to ensure continued development of HDF5 technologies, and the continued accessibility of data stored in HDF.
Platform layer	System Real Time Management	JMX	Java Management Extensions (JMX) is a Java technology that supplies tools for managing and monitoring applications, system objects, devices (e.g. printers) and service oriented networks. Those resources are represented by objects called MBeans (for Managed Bean). In the API, classes can be dynamically loaded and instantiated. Managing and monitoring applications can be designed and developed using the Java Dynamic Management Kit.
	Performance management	Nagios	Nagios is an open source computer system monitoring, network monitoring and infrastructure monitoring software application. Nagios offers monitoring and alerting services for servers, switches, applications, and services. It alerts the users when things go wrong and alerts them a second time when the problem has been resolved.

LAYER	FUNCTIONALITY	CHOICE	DESCRIPTION
	Version Control System	GIT	<p>Git is a distributed revision control and source code management (SCM) system with an emphasis on speed.</p> <p>Every Git working directory is a full-fledged repository with complete history and full version tracking capabilities, not dependent on network access or a central server.</p>
	Distributed file systems		
	NAS	Apache Hadoop HDFS	<p>The Hadoop distributed file system (HDFS) is a distributed, scalable, and portable file-system written in Java for the Hadoop framework. Each node in a Hadoop instance typically has a single namenode; a cluster of datanodes form the HDFS cluster. The situation is typical because each node does not require a datanode to be present. Each datanode serves up blocks of data over the network using a block protocol specific to HDFS. The file system uses the TCP/IP layer for communication. Clients use remote procedure call (RPC) to communicate between each other.</p>
	SAN	LVM on Lustre	<p>Logical volume management or LVM provides a method of allocating space on mass-storage devices that is more flexible than conventional partitioning schemes. In particular, a volume manager can concatenate, stripe together or otherwise combine partitions into larger virtual ones that administrators can re-size or move, potentially without interrupting system use.</p> <p>Volume management represents just one of many forms of storage virtualization; its implementation takes place in a layer in the device-driver stack of an OS (as opposed to within storage devices or in a network).</p> <p>Lustre is a type of parallel distributed file system, generally used for large-scale cluster computing. Lustre file system software provides high performance file systems for computer clusters ranging in size from small workgroup clusters to large-scale, multi-site clusters.</p>

LAYER	FUNCTIONALITY	CHOICE	DESCRIPTION
	Business continuity		
	Installation and Deployment	RPM	Red Hat Package Manager or RPM Package Manager (RPM) is a package management system. The name RPM variously refers to the .rpm file format, files in this format, software packaged in such files, and the package manager itself. RPM was intended primarily for Linux distributions; the file format is the baseline package format of the Linux Standard Base.
	Artifact Manager / Repository	Nexus	From [http://www.sonatype.org/nexus/], Sonatype Nexus sets the standard for repository management providing development teams with the ability to proxy remote repositories and share software artifacts.
	Configuration Management	Ansible	Ansible is an open-source software platform for configuring and managing computers. It combines multi-node software deployment, ad hoc task execution, and configuration management. Additionally, Ansible prefers to be categorized as an orchestration engine. It manages nodes over SSH and does not require any additional remote software (except Python 2.4 or later) to be installed on them. Modules work over JSON and standard output and can be written in any programming language. The system uses YAML to express reusable descriptions of systems.
	High Availability	Red Hat cluster suite	The Red Hat cluster suite includes software to create a high availability and load balancing cluster.
	Backup and Restore	LVM snapshots	LVM is commonly used for Performing consistent backups by taking snapshots of the logical volumes. A snapshot is the state of a system at a particular point in time. It can refer to an actual copy of the state of a system or to a capability provided by certain systems.
	Issue Tracking	Atlassian JIRA	JIRA is a proprietary issue tracking product, developed by Atlassian. It provides bug tracking, issue tracking, and project management functions.

LAYER	FUNCTIONALITY	CHOICE	DESCRIPTION
			JIRA is a commercial software product that can be licensed for running on-premises or available as a hosted application.
	OS	Red Hat Enterprise Linux	Red Hat Enterprise Linux (RHEL) is a Linux distribution developed by Red Hat and targeted toward the commercial market.
Infrastructure layer	VM management	virt-manager and libvirt	<p>The Red Hat Virtual Machine Manager is a desktop-driven virtual machine manager with which users can manage virtual machines (Vms).</p> <p>Libvirt is an open source API, daemon and management tool for managing platform virtualization. It can be used to manage KVM, Xen, VMware ESX, QEMU and other virtualization technologies. These APIs are widely used in the orchestration layer of hypervisors in the development of a cloud-based solution.</p>
	VM hypervisor	Qemu and KVM	<p>QEMU (short for "Quick EMULATOR") is a free and open-source hosted hypervisor that performs hardware virtualization (not to be confused with Hardware-assisted virtualization).</p> <p>QEMU is a hosted virtual machine monitor: It emulates central processing units through dynamic binary translation and provides a set of device models, enabling it to run a variety of unmodified guest operating systems. It also provides an accelerated mode for supporting a mixture of binary translation (for kernel code) and native execution (for user code), in the same fashion as VMware Workstation and VirtualBox do. QEMU can also be used purely for CPU emulation for user-level processes, allowing applications compiled for one architecture to be run on another.</p> <p>KVM (Kernel-based Virtual Machine) is a virtualization infrastructure for the Linux kernel which turns it into a hypervisor. KVM requires a processor with hardware virtualization extension.</p>
	OS	Red Hat Enterprise Linux	Red Hat Enterprise Linux (RHEL) is a Linux distribution developed by Red Hat and targeted toward the commercial market.

LAYER	FUNCTIONALITY	CHOICE	DESCRIPTION
	Hardware devices	N/A	[Specification of hardware devices are out of the scope of this project]
Security services layer	Authentication, User Management (single-sign-on)	Atlassian Crowd	<p>From [https://www.atlassian.com/software/crowd/overview], “Identity management for web apps. Finally, a single sign-on and user identity tool that's easy to use, administer, and integrate.</p> <p>Users can come from anywhere: Active Directory, LDAP, Crowd itself, or any mix thereof. Control permissions to all your applications in one place – Atlassian, Subversion, Google Apps, or your own apps.”</p>
	Authorization	Security JAAS	<p>Java Authentication and Authorization Service, or JAAS, is the Java implementation of the standard Pluggable Authentication Module (PAM) information security framework.</p> <p>The main goal of JAAS is to separate the concerns of user authentication so that they may be managed independently. While the former authentication mechanism contained information about where the code originated from and who signed that code, JAAS adds a marker about who runs the code. By extending the verification vectors JAAS extends the security architecture for Java applications that require authentication and authorization modules.</p>
	SIEM	Splunk	<p>Splunk is an American multinational corporation headquartered in San Francisco, California, which produces software for searching, monitoring, and analyzing machine-generated big data, via a web-style interface.</p> <p>Splunk (the product) captures, indexes and correlates real-time data in a searchable repository from which it can generate graphs, reports, alerts, dashboards and visualizations.</p> <p>Splunk aims to make machine data accessible across an organization and identifies data patterns, provides metrics, diagnoses problems and provides intelligence for business operation. Splunk is a horizontal technology used for application management, security and compliance, as well as business and web analytics.</p>

2.6.2 Network

Illustration 13 shows an overview of the different networks designed for the system.

There are three area networks:

- **Wide Area Network (WAN)** - A network that covers a broad area using leased telecommunication lines. In this case we use the same term for talking about international communications (Internet) and communications with in-orbit spacecraft.

This WAN would have the following external interfaces:

- A firewalled router with access to the internet, connecting with the EO user community;
- A firewalled router connected to the TC&TM ground station antenna, for monitoring and control of the satellite;
- A firewalled router connected to the instrument data ground station antenna, for reception of sensor data.

- **Local Area Network (LAN)** - A computer network that interconnects computers within a limited area (such as this data centre) using network media. The defining characteristics of LANs, in contrast to wide area networks (WANs), include their smaller geographic area, and non-inclusion of leased telecommunication lines. In this case we define two internal LANs (sub-networks) with different bandwidth and latency capabilities:

- **Management network** – Connecting all hardware devices of the system, used for management and messaging data traffic. This network includes point-of-access switches and core switches in a star-shaped configuration. VM and processing management nodes and other auxiliary devices are part of this network. The WAN access to Internet is directly linked to the core switch of this sub-network.
- **Fast data network** – Connecting only VM host nodes and the Storage Area Network, and the Processing nodes, used for fast movement of large data objects. This network includes point-of-access switches and core switches in a star-shaped configuration. Both WAN accesses to the Ground Antennas are directly linked to the core switch of this sub-network.

Illustration 14 gives a detailed view of those star-shaped network, connected devices, interfaces (Network Interface Controllers, NICs) installed on each node and the proportionated bandwidths of each connection (shown with different line thicknesses).

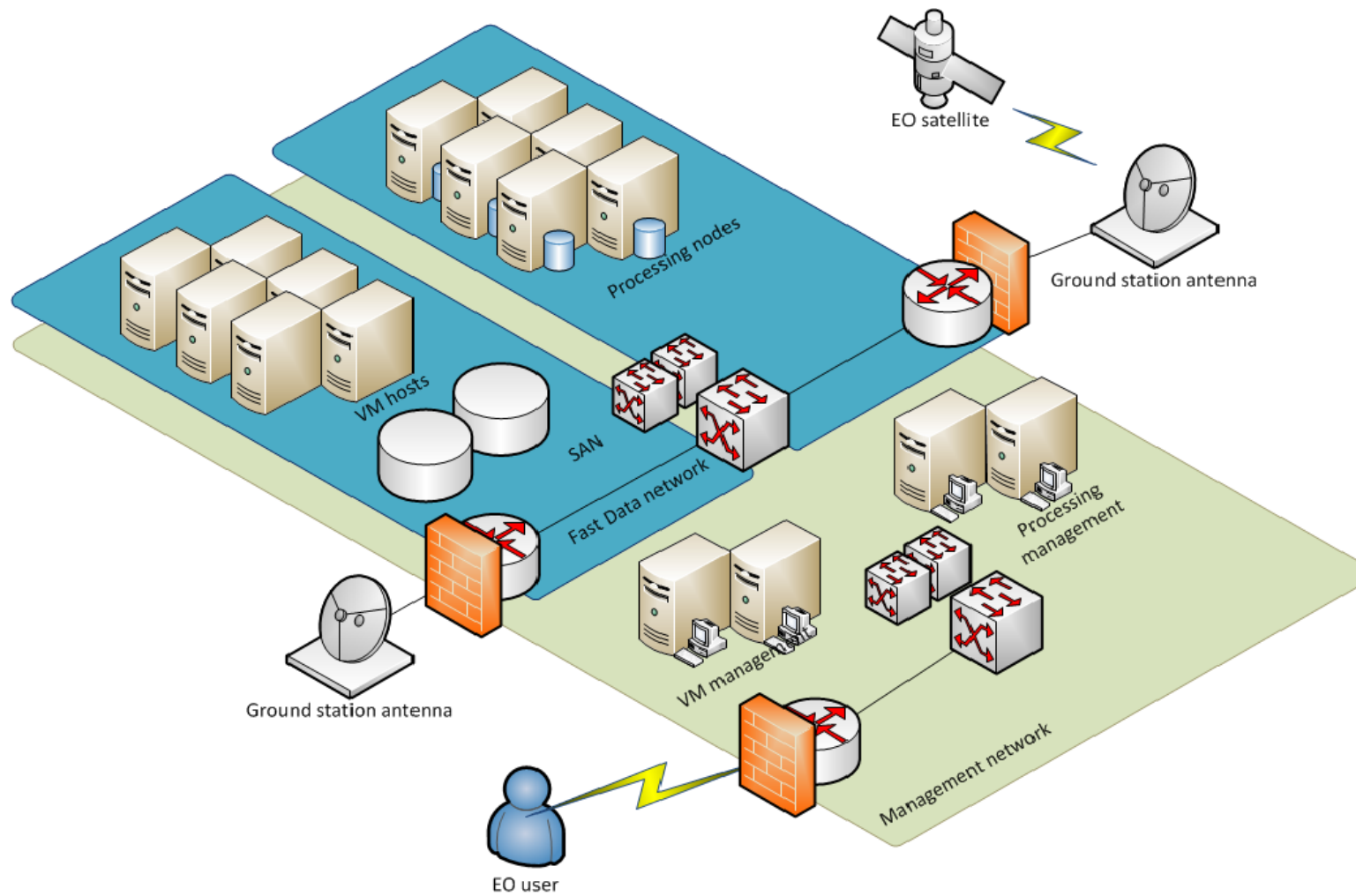


Illustration 13: Network diagram

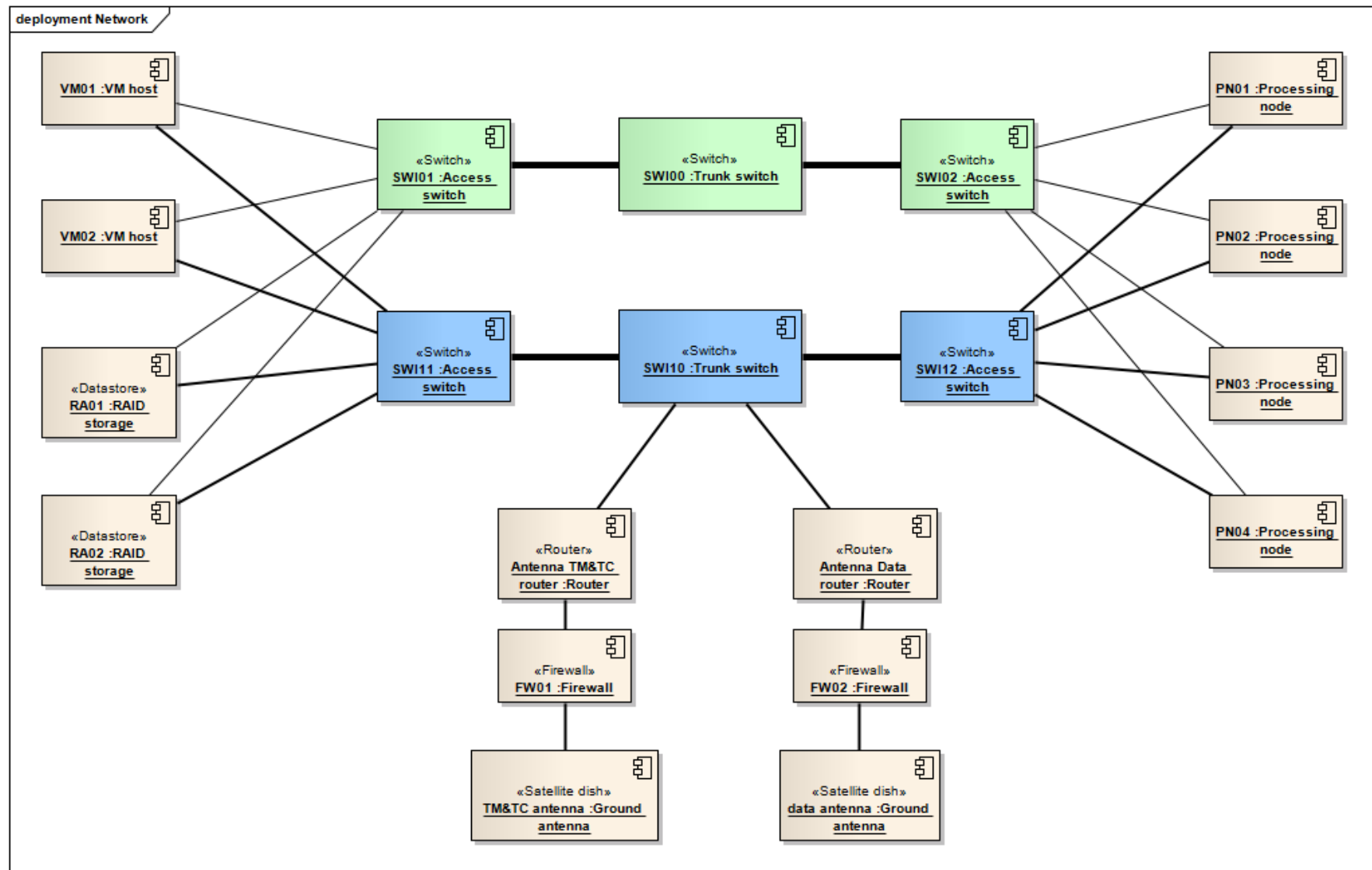


Illustration 14: Network detailed diagram

2.6.3 HARDWARE

Without a system sizing exercise, that is, without knowing the data size budget, retention policies and performance metrics, it is impossible to give a clear specification of the physical resources needed for providing a functioning system.

This section and Illustration 15 just give an approximated solution to this generic, small sized system. A real system should be dimensioned accordingly to the estimated needs.

The system would be deployed in four racks:

- Two VM racks, hosting the VM nodes and Storage Area Networks;
- One processing racks, keeping all processing nodes (each with internal storage);
- One half-rack with management nodes, WAN routers and firewalls, and I/O devices (tape library).

Both VM racks and the processing rack are connected to the VM management rack via inter-rack cabling, for both management and fast data networks.

Each rack would have:

- An access switch with intra-rack connections with the rest of the devices, for both management and fast data networks;
- A KVM switch connecting all suitable nodes with a common Keyboard, Video monitor and Mouse;
- Uninterrupted Power Supplies, temperature sensors, cooling systems and cable organizers;

The room should comply with environmental, security and safety requirements and comply with current legislations on disasters and personnel management.

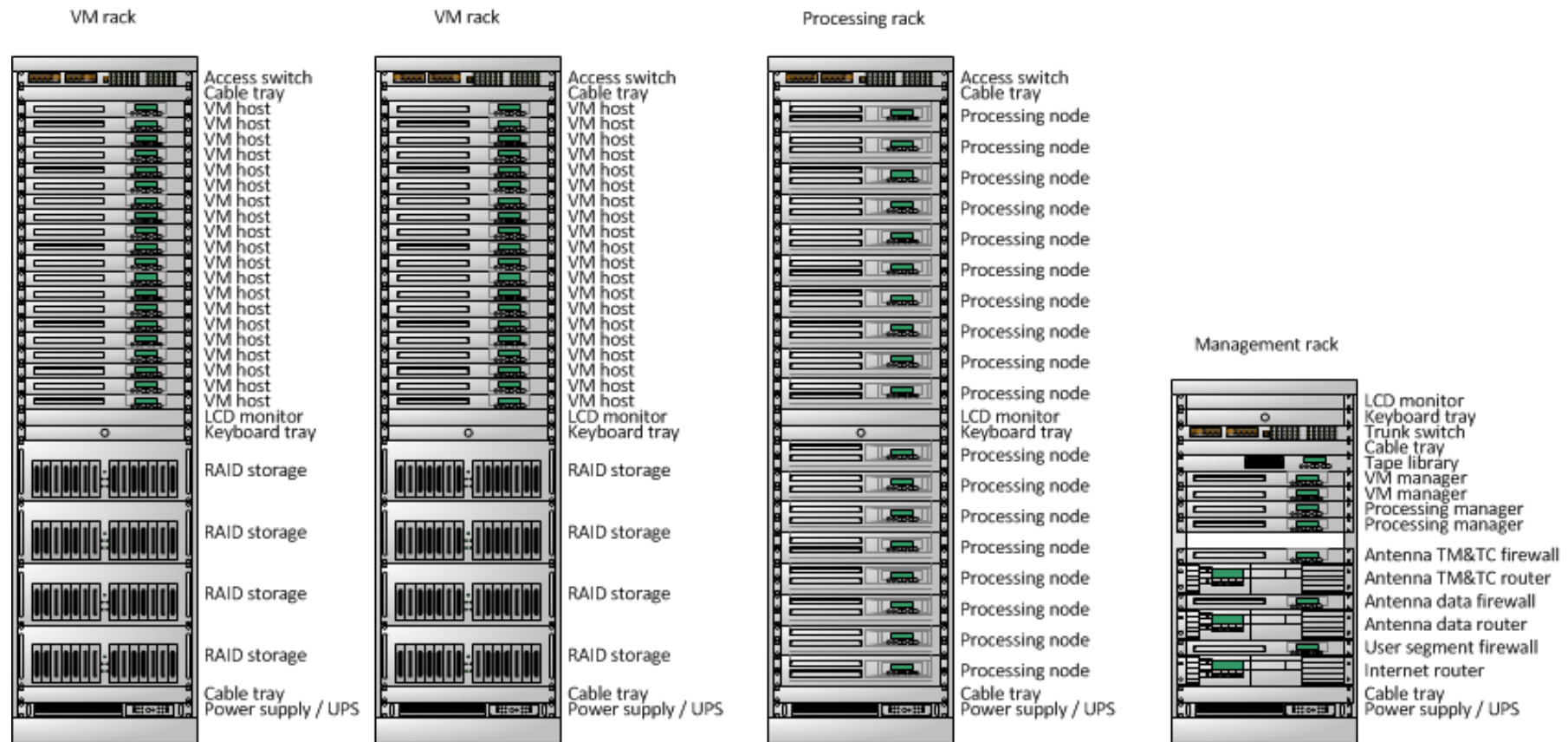


Illustration 15: Physical deployment

3 CONCLUSIONS

This section lists the extracted conclusions of this work and possible future evolutions and lines of work.

This project has proposed an integral, multi-disciplinary architectural design that has reached the objectives defined initially:

- Update essential systems – All proposed technologies listed in the Technological Viewpoint are state-of-the-art, proven applications, tools and languages;
- To enhance the classical production cycle – This system proposes the use of common services, common messaging buses, common platforms and a bunch of other common functionalities (as shown in the Computational Viewpoint) that would eliminate redundant, unnecessary developments;
- To enable cost and risk reductions – Again, the component-based, layered architecture described in the Computational Viewpoint would reduce cost and risks. For instance, the software technology proposed (beside the space-related functions) would cost only a few hundreds of Euros, compared with the hundreds of thousands for a typical small scale development;
- To give a reference architecture framework for Earth Observation ground systems – This system is generic, scalable and adaptable enough to be used for other EO ground systems or, at least, serve as inspiration for other methods of work.

Once the basis of this concept are proven, the project might evolve in any of these directions:

- Provide a mock-up implementation of this system, using functional skeletons, simple business rules and virtualized infrastructure;
- Propose alternative, distributed deployments of the instances of part of the system, to better depict real life cases in where, for instance, it exists several satellites, ground stations spread around the world with different track and control capabilities, local data processing stations or data distribution points on data centers or the cloud;
- Describe more in detail the different activities in the development lifecycle including test, training and pre-production environments and the validation and verification processes;
- Include the rest of the space-related functionalities not covered in this project, like the platform and payload simulators, the key management facilities (for security procedures) or the long term archives;
- Describe more in detail how the space-related functions would work with and benefit of the presented architecture, now that much of its functionality is taken off of them and some new technologies and methods are proposed.

Whichever path is taken, it is guaranteed that it would be an extra added-value to this project.

4 GLOSSARY

The following definitions are adapted from (LIRD) and provided here to clarify requirements using the defined terms.

- Level 0: Raw data reconstructed to unprocessed instrument data at full space-time resolution with all available supplemental information to be used in subsequent processing (e.g. ephemeris, health and safety) appended.
- Level 1a: Unpacked, reformatted and resampled Level 0 data with all supplemental information to be used in subsequent processing appended. Data generally presented as full space/time resolution.

A wide variety of sub-level products are possible.

- Level 1b data: Unpacked, reformatted, and resampled Level 0 data with all supplemental information to be used in subsequent processing appended. Radiometric and geometric correction applied to produce parameters in physical units. Data generally presented as full space/time resolution. (This is identical to the earth-referenced instrument data with radiometric calibration applied and all calibration data appended).
- Level 2: Retrieved environmental variables (e.g. sea surface temperature) at the same resolution and location as the Level 1 source.
- Level 2+: All Level 2 and higher products
- Level 3: Data or retrieved environmental variables which have been spatially and/or temporally resampled (i.e. derived from Level 1 or 2). Such resampling may include averaging and compositing.
- Level 4: Model output or results from analyses of lower level data (i.e. variable that is not directly measured by the instruments, but are derived from these measurements).
- Mean Time between Failures: the average time that a system/component that works without failure
Mean Time to Failure: the expected time that a system/component will operate before the first failure will occur.
- Mean Time to Repair: the average time required to repair a system/component.
- Metadata is non-radiometric data that provides additional information on the data collection conditions including latitude and longitude information, day, time, data quality flags that depend on the level (0, 1, 1b, 2, etc.) of the data associated with the metadata, and any additional space-ground ICD required information.
- Mission availability is the probability that the entire EO system can be successfully used for its specified mission over the stated period of time.
- Operational lifetime of the EO satellite begins immediately after instrument check-out of the satellite on-orbit and extends through the operational usage of the EO satellite while meeting the mission availability requirements.

- Raw Data: X-band data (instrument and some telemetry data) in their original packets, as received from the satellite.
- Satellite consists of a spacecraft to support the instruments, the instruments, the associated communication systems, and the communications payload services.
- Single point failure is a failure of a hardware or software element with no redundancy.
- Spacecraft is a vehicle without instruments, but including the magnetometer and the raw data downlink satellite service, propulsion system, power system, thermal system, GN&C, and structure, that is intended to be launched into space by a launch vehicle.
- User Community is a general term describing the aggregate of EO users.

The following abbreviations are used throughout the document:

- EO – Earth Observation
- EM, EMS – Enterprise Management, Enterprise Management Segment
- FO, FOS – Flight Operations, Flight Operations Segment
- GNCC – Guidance, Navigation and Control
- GS – Ground Segment
- ICD – Interface Control Document
- PDG, PDGS – Product Data Generation, Product Data Generation Segment
- PD, PDS – Product Distribution, Product Distribution Segment
- TBD – To Be Defined

5 BIBLIOGRAPHY

CCSDS. (2008). *Reference Architecture for Space Data Systems* (Issue 1 ed.). Washington DC, USA: CCSDS Secretariat.

CCSDS. (2013). *Reference Architecture for Space Information Management*. Washington DC: CCSDS Secretariat.

Euroconsult. (2008). *Satellite-Based Earth Observation, Market Prospects to 2017*. Paris: Euroconsult.

NASA. (2012). *GOES-R Series Level I Requirements (LIRD)*.

SkyBox Imaging. (2014, June 10). *News and Events*. Retrieved from www.skyboximaging.com:
<http://www.skyboximaging.com/blog/skybox-imaging-google>

*To Yolanda, my beloved wife, who has taught me that Time is not a foe but a friend,
and patience, constancy and effort are also ways to reach success.*

In England, Summer of 2014: almost twenty years after I started my studies.



José Julio Ramos Pérez, A.K.A. "Slowpoke"