"SPECIFICATION AND ARCHITECTURE OF A SYSTEM FACTORY FOR SPACE SYSTEMS"

Tiago Jorge [(1)], Marina García [(1)], Carlos Redondo [(1)], Clement Goujon [(2)], Mauro Pasquinelli [(2)], Stephan Jahnke [(3)], Riccardo Benvenuto[(3)], Jean-Baptiste Bernaudin [(4)], Pascal Roques [(5)], Claire Parfitt [(6)], Andreas Jung [(6)], Elena Alaña [(1)]

(1) GMV Aerospace and Defence, E-mail:tiago.jorge@gmv.com, marina.garcia.d@gmv.com, carlos.redondo.aparicio@gmv.com, ealana@gmv.com

(2) Thales Alenia Space Italy, E-mail: clement.goujon@thalesaleniaspace.com, mauro.pasquinelli@thalesaleniaspace.com

(3) OHB System AG, E-mail: stephan.jahnke@ohb.de, riccardo.benvenuto@ohb.de

(4) Airbus Defence and Space SAS, E-mail: jean-baptiste.bernaudin@airbus.com

(5) PRFC, E-mail: pascal.roques@prfc.fr

(6) European Space Agency, E-mail: claire.parfitt@esa.int, andreas.jung@esa.int

Abstract: The deployment of Model-Based Systems Engineering (MBSE) in space projects is not straightforward. The interactions among stakeholders at various levels happen to be difficult because the tools involved are not fully interoperable. One of the key elements that would facilitate and ensure the exchange of engineering data information, is the definition of a Systems Engineering supporting infrastructure, also called *System Factory*, that would allow implementing this interoperability. This paper introduces the Operational Analysis, System Need Analysis and Logical Architecture of the System Factory designed following the *ARCADIA method* [1] and using the Capella tool. This paper is the result of the SASyF activity ("*Specification and Architecture of a System Factory*").

Keywords: Arcadia, Capella, Data Hub, MBSE, Ontology, System Factory.

1. INTRODUCTION

The objective of the SASyF project is to define the specification and architecture of a Model-Based Systems Engineering infrastructure for Space Systems Engineering, the so-called *System Factory*, covering all phases of a space system development, by applying the *ARCADIA method*. In particular, we model how a System Factory supports the system engineers in executing the tasks described in the standard ECSS-E-ST-10 [2], following a MBSE approach. Currently, no disciplines are modelled.

This activity involves 3 Large Satellite Integrators (LSIs) / Primes: *Airbus*, *Thales Alenia Space* and *OHB*, that provide their knowhow and expertise to establish the architecture of the System Factory. Additionally, Pascal Roques, as expert on Arcadia and Capella, provides guidance to ensure the correct modelling of the System Factory.

The scope is a local System Factory, e.g. a System Factory at an LSI, or it could be tailored to the one of an Agency or an LSI's subcontractor.

A special focus is put on information exchanged between the different stakeholders, the capabilities supported by the factory and the internal interactions occurring within that infrastructure.

The modelling started with the definition of 9 core use cases that compile the main activities that shall be performed in a Space System development including all relevant stakeholders and the complete development process. In order to fit the System Factory with the current activities required to develop Space Systems, the use cases were provided by the LSIs based on their experience on building such systems. The use cases have been defined and derived from ECSS-E-ST-10 [2] and were organised according to the space Systems Engineering activities when a model-based approach is adopted. Namely, there is one use case per main Systems Engineering activity (these use cases integrate several sub use cases):

Use cases: *Requirements engineering; Analysis; Design and configuration; Verification; Management and planning; Interface control; Design files production; Risk management; Support to configuration control, change management and NC control.*

2. SYSTEM FACTORY MODEL

2.1. OVERVIEW

Figure 1 presents an overview of the information included in the Capella model at each Arcadia level. This figure facilitates the identification of relationships among the different types of elements and in particular the link between levels. The physical elements are not yet modelled but are in the image represented (in orange) since the

next step is to map Logical Components to concrete tooling, i.e. the Physical Architecture.

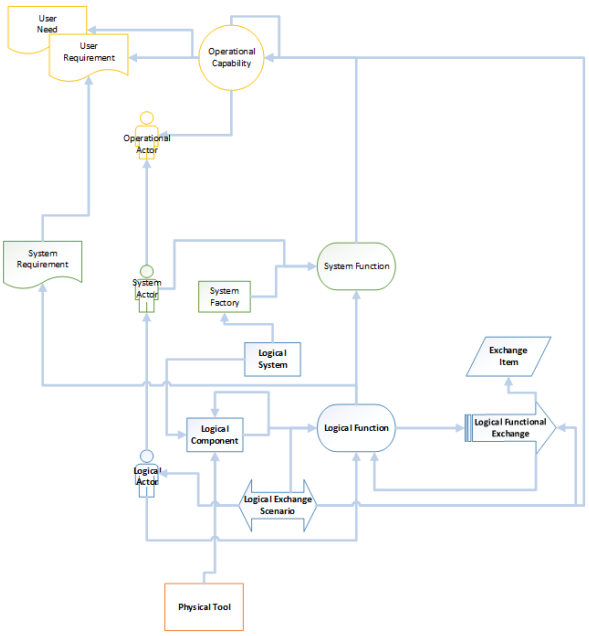

Figure 1: Overview of the Capella model
Legend: Elements at Operational Analysis are represented in yellow; System Need Analysis in green; Logical Architecture in blue; and Physical Architecture in orange

Traceability is maintained in the model between all these levels (e.g. traceability matrix between the Logical Functions and the System Requirements and vice versa).

In this paper we use the *Requirements Engineering use case* as example for the various partial illustrations.

2.2. OPERATIONAL ANALYSIS

The operational analysis was the first Arcadia step to be performed. It is simple and represents a high-level view of the user needs [3], identifying the main capabilities and actors.

Here the *User Requirements* are modelled and traced to the *sub-use cases* (modelled as refined *capabilities*). The User Requirements are then traced to the System Requirements.

USE CASE TITLE	Requirements Engineering	PHASES	Mostly (but not limited to) early phases (A/B).
DESCRIPTION	Main sub-use cases: Requirements engineering is not a single pass and it is performed by iterations with the maturity increase from both *Customer* and *Supplier* sides. Thus the need to have a tool to share easily both customer requirements and technical specifications with history and impact tracking is needed. 1.1. Customer requirements analysis (phases A/B) As a *Customer* I need to deliver the specifications to my supplier, and being able to receive comments, compliance status and change proposals, and to perform co-engineering to support the improvement of the requirements specification during early project phases and to evaluate the impact of any proposed changes. As a *Supplier*, I need to receive the customer specification and related ancillary specifications (Level 0, L-0 spec: SRD, IRD, etc.) in a format compatible with my requirements management system and being able to separate internal comments with customer-related comments, and being able to iterate with the customer so to improve the common understanding in its requirements specification. As a *Project Manager* or *Systems Engineer* of the *Customer* or the *Supplier* I need to keep control of such flow of information and approve such exchanges. (Applicable to all sub use cases). *Exchanges: Specifications from customer (MRD, SRD, IRD, etc.), supplier comments and change proposals, compliance matrices*		

Figure 2: 'Requirements Engineering' – description of sub-use 'Customer requirements analysis'

The User Requirements represent the user needs for a typical space system development process from different users' perspectives. Therefore, they are user-oriented and are derived from the Operational Analysis use cases.

Requirement Id	Requirement Text	Trace to Sub-Use Cases
URD-REQ-1010	The customer shall be able to deliver its specifications to the supplier.	UC 1.1
URD-REQ-1020	The customer specifications shall be delivered in a format which can be imported by the supplier.	UC 1.1
URD-REQ-1030	It shall be possible to exchange comments and related answers on customer specifications between customer and supplier.	UC 1.1
URD-REQ-1040	The supplier shall be able to provide to the customer the state of compliance w.r.t. to the provided specifications and applicable documents in matrix form or via compliance statements.	UC 1.1 UC 1.2
URD-REQ-1050	It shall be possible to plan, execute and trace co-engineering sessions between customer and supplier, to improve understanding and formulation of customer specification or of supplier specifications.	UC 1.2

Figure 3: Part of the User Requirements for the Requirements Engineering use case

The roles have been simplified/abstracted to provide an Operational Analysis which is not dependent from the type of discipline involved in the project, simplifying also the modelling effort.

2.3. SYSTEM NEED ANALYSIS

The System Need Analysis level derives concrete information from the Operational Analysis level, detailing the *scope* of the System Factory, including the interfaces with external actors.

Here the architecture of the System Factory is defined as a *"black box"* and its *boundary* is identified together with the *functions* handled either by the System Factory or by external actors. The criteria to determine if a functionality is performed or not by the System Factory is limited by the fact that the proposed System Factory is defined at *company level*. Therefore, the interaction with other similar infrastructures, e.g. in the customer side, are represented as exchanges with the corresponding actors.

The System Requirements are modelled and traced up to User Requirements (Operational Analysis) and down to Logical Functions (Logical Architecture – see next section). The System Requirements are derived from the User Requirements considering the scope of the System Factory. Therefore, they specify if the user needs are satisfied by the System Factory itself. These are divided in groups, namely:

Functional requirements: *Requirements specification; System modelling; Analysis; Verification; Configuration control; Interfaces; Management and planning.*

Non-Functional requirements: *Performance requirements; Availability; Design and implementation constraints; Usability; Security; Model obsolescence management.*

Requirement Id	Requirement Text
SYS-REQ-FUN-0010	The *system factory* shall allow delivering customer specifications to different entities depending on the industrial set-up.
SYS-REQ-FUN-0020	The *system factory* shall allow receiving supplier specifications from a non-model-based approach. *NOTE: The required flexibility of the system factory to import different types of format and the ability to provide a framework to develop import scripts will be developed in the lower-levels.*
SYS-REQ-FUN-0030	The *system factory* shall allow importing customer specifications and integrate it as requirements into the model. *NOTE: Customer requirements are not only mission-level requirements, but can also refer to system/sub-system requirements or via an applicability matrix (e.g. environment requirements).*
SYS-REQ-FUN-0040	The *system factory* shall allow storage and editing of specifications and requirements.
SYS-REQ-FUN-0050	The *system factory* shall manage structure specifications.
SYS-REQ-FUN-0060	The *system factory* shall allow allocation of specification sections /requirements.
SYS-REQ-FUN-0070	The *system factory* shall store and provide reference requirements.
SYS-REQ-FUN-0080	The *system factory* shall allow tailoring of standard requirements libraries and related application to projects or class of projects.

Figure 4: System functional requirements for requirements specification

2.4. LOGICAL ARCHITECTURE

2.4.1. Characteristics

The Logical Architecture is the *main output* and presents *how* the system works to fulfil expectations. The level of Logical Architecture aims to identify *Logical Components* inside the System, their relations and their contents, *independently of any considerations of technology or implementation*.

The Logical Architecture ensures completeness of *Functions* and *Exchange Items* necessary to establish interoperability across industry and agencies. Together with the Space System Ontology (OSMoSE activity [4]), it will facilitate the interoperability due to the common interfaces and common semantics. It will mainly be used by Primes, and ESA will use it to interface with Primes in order to have smoother interactions. It allows to have a common way to map their own architectures and define standard interfaces.

There is not a unique logical solution, however the resulting SASyF Logical Architecture shall be a *reference point* for all companies to implement their Physical Architectures and it represents one feasible alternative already agreed by the 3 LSIs.

The focus is put on what is exchanged between stakeholders and components. It is nevertheless still a high-level abstract architecture that should evolve to be more precise when new digital engineering practices are clarified.

The proposed Logical Architecture is based on *high-level common automation needs*. This was the main driver of the architecture. The most common needs related to automation were extracted from the Use Cases and User Requirements descriptions, and used to *drive the design of the architecture*. This was done together with a revisit to high-level User Needs [3] and general factory requirements. Consequently, many Logical Components were derived directly from these automation needs - e.g. *component Configuration Manager* covering the configuration management needs. This was an opportunity to reduce, "by construction", potential redundancies in the architecture, while, at the same time, to extract and therefore meet the most relevant needs (based on the premise: if they are common, they must be relevant).

2.4.2. Diagrams

The model goes beyond defining just an architecture to also define main functional chains and exchange scenarios. The model includes the following Capella diagram types at logical level:

Logical Class Diagram Blank (LCDB): It shows the main artefacts exchanged, i.e. the Exchange Items. Each Exchange Item includes a description (tab Description in the Properties view) indicating what information it covers.

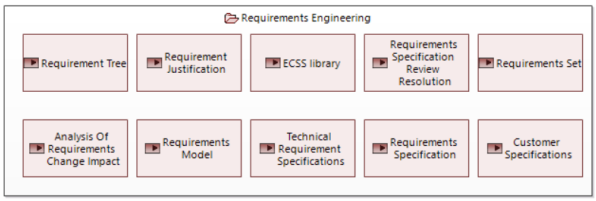

Figure 5: LCDB for Requirements Engineering

Logical Exchange Scenario (LES): It shows a sequence of Logical Functional Exchanges instances (horizontal lines) representing a scenario of interest and consistent with the architecture design. The Functional Exchanges can be linked with the respective Exchange Items. The Logical Actors and Components participating in the scenario are shown as vertical lines.

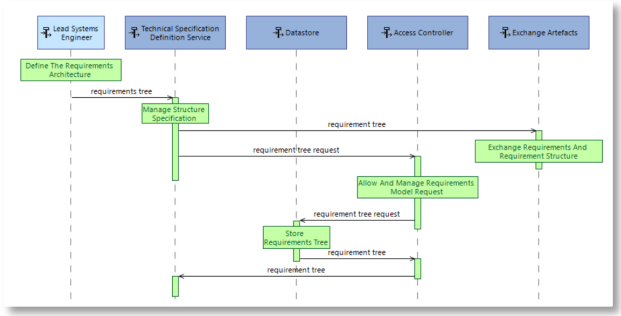

Figure 6: LES for defining and providing to the Customer the Requirements Architecture

Logical Functional Chain Description (LFCD): It shows a chain (composition) of Functional Exchanges passing through the Logical Functions

to which they are connected as input or output. It represents an individual data flow (whereas a LES can present/exercise multiple of such possible data flows).

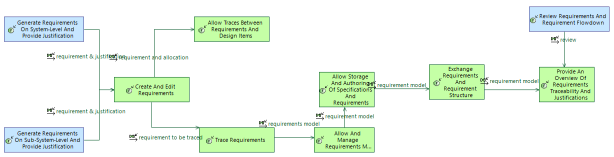

Figure 7: LFCD for providing evidence to the Customer of the requirements flow-down and their justification

Logical Architecture Blank (LAB): It shows the Factory Logical Components breakdown / organisation. The Logical Functions are allocated to those components or Logical Actors and are connected via Logical Functional Exchanges, which in turn can be linked with the respective Exchange Items. Notice the usage of Functional Chains (in blue/red) to mark the main data flows.

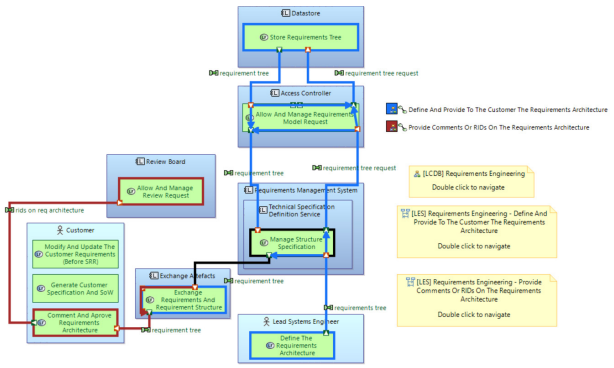

Figure 8: LAB for Technical Specification Definition

Note that one can use the Capella views *Properties* and in particular the *Semantic Browser* to explore the model and navigate between model elements and diagrams.

2.4.3. Metrics

Main current model metrics are: *Logical Components: 96; Logical Functions: 445; Exchange Items: 111; Functional Exchanges: 668; Scenarios: 46.*

Effort related metrics include: *22 Working sessions (2 hours each every 2 weeks); 27 contributions (Capella model from LSIs); 6 organisations involved (ESA, GMV, TAS, ADS, OHB, PRFC); ~3-4 people involved per organisation; 495 model repository commits.*

2.4.4. Data Hub

The *Data Hub* is the main and most central Logical Component, as most of the other Logical Components interact with/through it.

The definition, maintenance and exchange of the different Exchange Items (artefacts) is performed mainly as models and via a central and unique source of truth: the Data Hub Logical Component.

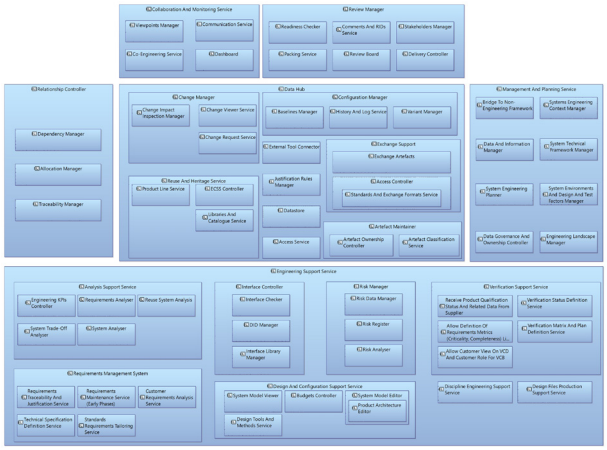

Figure 9: Main Logical Components including the Data Hub

3. NEXT STEPS

Currently we are working on the mapping of the Logical Architecture onto the Physical Architecture. This will also allow to perform a gap analysis, identifying to which level the LSIs have already implemented the Logical Architecture in their organizations. An objective is to identify tools that are closely connected (e.g. model editor, configuration management tool) and shall accordingly be interoperable. Ultimately, we will refine the Logical Architecture based on feedback when the LSIs' Physical Architectures are modelled.

Importantly this work is reviewed also by ESA's MB4SE steering group [5], to receive feedback from the wider community for the improvement of the model.

4. REFERENCES

[1] ARCADIA method: https://www.eclipse.org/capella/arcadia.html

[2] "Space Engineering – System engineering general requirements", ECSS-E-ST-10C Rev.1, 15 February 2017.

[3] "MB4SE User Needs", MB4SE-TN-001, Issue 2, Rev. 2.

[4] OSMoSE: https://mb4se.esa.int/OSMOSE_Main.html

[5] MB4SE: https://mb4se.esa.int/

5. ACKNOWLEDEMENTS

This work was supported by the SASyF project, ESA contract 4000129181/19/NL/AS. The partners involved in this study are GMV as prime contractor and Thales Alenia Space Italy, Airbus Defence and Space SAS and OHB System AG as subcontractors, and the support of Pascal Roques (PRFC) as External Consultant.