DELIVERABLE D 2.3

AGILE4.0 PARADIGM ARCHITECTURE & PROCESSES

Author(s): DLR
WorkPackage N°: WP2
Due date of deliverable: 31.08.2021
Actual submission date: 30.11.2022
Document ID: D2.3 - AGILE4.0 Paradigm Architecture Processes - v05.docx

Grant Agreement number: 815122
Project acronym: AGILE 4.0
Project title: Towards cyber-physical collaborative aircraft development
Start date of the project: 01/09/2019
Duration: 42 months

Project coordinator name & organisation:
Luca Boggero
DLR – System Architectures in Aeronautics | Aircraft Design and System Integration
Tel: +49 40 2489641-338
E-mail: Luca.Boggero@dlr.de

Project website address: www.agile4.eu

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON 2020 RESEARCH AND INNOVATION FRAMEWORK PROGRAMME UNDER GRANT AGREEMENT NO 815122

DOCUMENT INFORMATION

Document ID	D2.3 - AGILE4.0 Paradigm Architecture Processes - v05.docx		
Version	0.5		
Version Date	30.11.2022		
Author	Luca Boggero (DLR)		
Dissemination level	PU		

APPROVALS

	Name	Company	Date	Visa
Coordinator	Luca Boggero	DLR	Nov 30, 2022	
WP Leader	Luca Boggero	DLR	Nov 30, 2022	
Reviewer	Susan Liscouët-Hanke	CONU	Aug 31, 2022	Nov 30, 2022

DOCUMENTS HISTORY

Version	Date	Modification	Authors
0.1	27.03.2022	Document initialization	Luca Boggero (DLR)
0.2	12.07.2022	Document is concluded and sent out for review	Luca Boggero (DLR)
0.3	05.11.2022	Document is modified after review	Luca Boggero (DLR)
0.4	11.11.2022	Document is finalized	Luca Boggero (DLR)
0.5	30.11.2022	Document is improved and ready for signatures	Luca Boggero (DLR)

LIST OF AUTHORS

Full Name	Organisation
Luca Boggero	DLR

DISTRIBUTION LIST

Full Name	Organisation
AGILE 4.0 Consortium	
Externals	

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

GLOSSARY

Acronym	Signification
ADSG	Architecture Design Space Graph
MBSE	Model Based Systems Engineering
MDAO	Multidisciplinary Design Analysis and Optimization
QoI	Quantity of Interest
SysML	System Modeling Language
WP	Work Package

1 EXECUTIVE SUMMARY

1.1 Introduction

The present deliverable is of the type "other", and it addresses the intermediate stage of a Systems Engineering Product Development process, which aims at designing and optimizing a system starting from identifying stakeholders and needs and including other activities such as requirements engineering. More specifically, D2.3 focuses on the *System Architecting* phase of the development process, which encompasses identifying functions to be performed by the system, allocating functions to different logical/physical components, and generating instances of system architecture. The deliverable presents the guideline and the process defined in AGILE 4.0 project for the system architecting. The deliverable D2.3 is publicly available, and the produced guideline is shared through the public link of the AGILE 4.0 Cloud: https://www.agile4.eu/cloud/index.php/s/xSqfgaja7R3YjxX.

1.2 Brief description of the work performed and results achieved

The research activities documented in this deliverable include a literature survey on system architecting, and investigation on Systems Engineering processes and Architectural Frameworks.

A first version of the process and guideline for system architecting developed in the project has been presented to the AGILE4.0 Consortium at the M18 during a virtual workshop. The process and guideline have been applied by the partners for the development of the seven Application Cases, and feedback has been collected. Following, the process and guideline have been revised and are now modeled and documented in the present report. Additional virtual workshops have been organized and contributed to refining and improving the work, eventually bringing to the results collected in the current version of the deliverable.

1.3 Deviation from the original objectives

1.3.1 Description of the deviation

The activities in preparation of the present deliverable were originally planned to start in September, 2020. The actual work started in January 2021 due to some delays in the project activities and the late start of WP2. In addition, due to the COVID-19 crisis, planned physical meetings had to be turned into web meetings, negatively impacting the progress.

1.3.2 Corrective actions

Apart from turning physical meetings into web meetings and shifting the deadline of actions and the deliverable D2.3 forward in time, no corrective actions were required.

2 INTRODUCTION

2.1 Aim of the deliverable and organization of the report

The deliverable D2.3 presents the process and guidelines for system architecting activities. Within the context of the AGILE 4.0 project, *system architecting* is defined as the identification of alternative configurations of the system. These alternative configurations consist of different components that aim at fulfilling the functions that the system should provide. This deliverable aims to provide the process for generating and representing system architectures.

The *System Architecting* phase represents the intermediated part of the Model-Based Systems Engineering (MBSE) process being set up within the AGILE 4.0 project for the development of systems. The MBSE process also includes the *Application Case Definition* phase (see deliverable D1.3 [1]) and the *Multidisciplinary Design Analysis and Optimization (MDAO)* of systems. This process is schematized in Fig. 1.

Fig. 1: Schema of the MBSE process addressed in AGILE 4.0 project.

This process starts with identifying system stakeholders and collecting their needs. Then, Concepts of Operations (ConOps) are elaborated to describe through operational scenarios how the system will operate during its life cycle and, therefore, to refine and validate the stakeholder needs. Validated needs are afterwards transformed into requirements, which drive the system architecting and its development. In other words, several potential solutions are defined by generating various system architectures consisting of different logical components. The system architectures are finally designed and optimized through MDAO processes. This deliverable focuses on the third and fourth steps of the proposed MBSE process: *System Architecting* and *System Synthesis*.

The present report describes the model of the process, ontology and viewpoints for system architecting. It is organized as follows. Chapter 2 introduces the topic of the deliverable, and highlights the relations between D2.3 and the other research activities of the project. Chapter 3 presents the MBSE Architectural Framework being developed in AGILE 4.0, providing the definitions of *Ontology*, *Viewpoint* and *Process*. The *Ontology* that includes the concepts of system architecting is described in Chapter 4. Chapter 5 collects and describes the *Viewpoints* identified for the representation of system architectures. The system architecting process is instead addressed in Chapter 6. Finally, the report presents conclusions and further remarks in Chapter 7.

The guidelines of deliverable D2.3 are made publicly available through diagrams in Diagram.net. The diagrams can be downloaded from: https://www.agile4.eu/cloud/index.php/s/xSqfgaja7R3YjxX.

2.2 Deliverable in the context of AGILE 4.0 and organization of the research tasks

Fig. 2 shows the main relations between the present deliverable and the other activities of the AGILE 4.0 project.

The activities of deliverable D2.3 are driven by some of the needs identified in deliverable D1.2 [2], more specifically stating the lack in the project of a guideline for system architecting. The derived system architecting guideline has then been explained to the partners of the Application Cases, and used for the preparation of deliverables D6.4 [3], D7.4 [4] and D8.4 [5]. These deliverables addressed the system architecting activities of the seven Application Cases by adopting a model-based approach. Then, feedback was collected from Work Packages WP 6, 7 and 8. This feedback was furtherly elaborated and a new guideline was defined for D2.3 and used in preparation of deliverable D4.6 [6] of WP4, which is about a model-based approach for system architecting. In other words, D4.6 presents the types of model that should be used in AGILE 4.0 for the representation of system architectures.

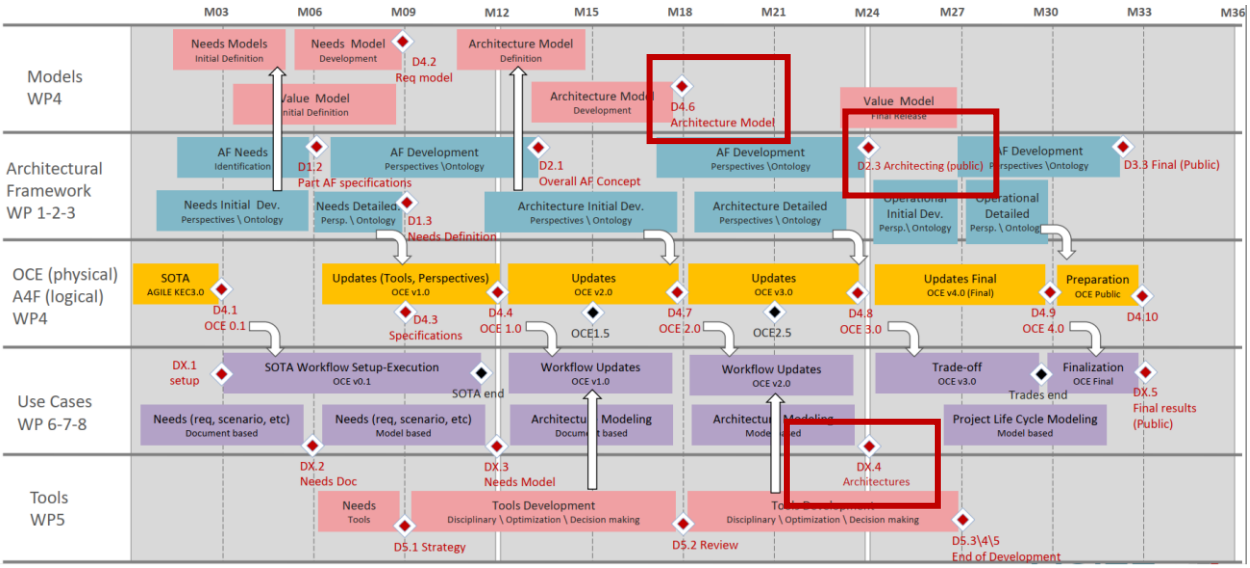


Fig. 2: Main deliverables and tasks of AGILE 4.0 project.

3 AGILE 4.0 MBSE ARCHITECTURAL FRAMEWORK

One of the goals of the AGILE 4.0 project is the definition of a new MBSE Architectural Framework. The ISO/IEC/IEEE 42010 standard defines an architectural framework as a set of:

"..conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders." [7]

In other words, an architectural framework is a guideline that can be used to represent system architectures. An architecture is a formal description of a system, its behaviour and the relationships among all the entities composing the system.

The MBSE Architectural Framework developed in AGILE 4.0 is structured in four layers, as shown in Fig. 3.

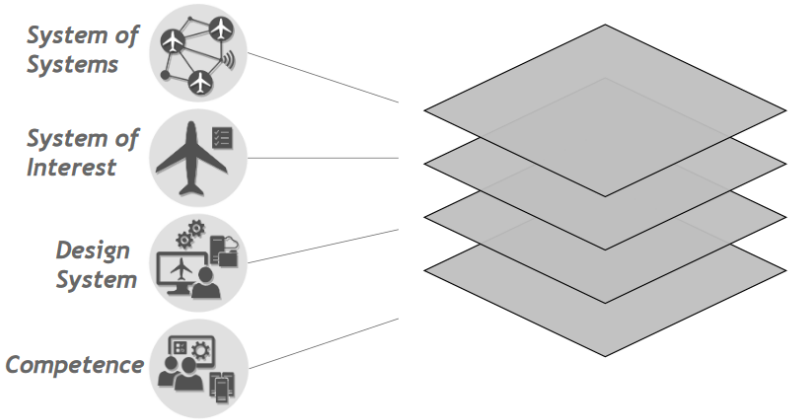


Fig. 3: Four-layer structure of the MBSE Architectural Framework addressed in AGILE 4.0 project.

One layer focuses on the *System of Interest*, which is the system being designed and optimized. The System of Interest can be for example an aircraft, a subsystem (e.g. ATA 24 – Electrical System) or a component (e.g. an electric generator), but also an enabling system, like a Supply Chain or a production system. The MBSE approach schematized in Fig. 1 is employed for the development of the System of Interest. In AGILE 4.0 project, seven Systems Of Interest are developed in WP6-7-8.

The System of Interest can be developed through a *Design System*, which is represented by a different layer. This layer encompasses all the Systems Engineering and MDAO processes required to develop the System of Interest. Again, an MBSE approach can be employed for the definition, architecting and development of this layer.

The development of a System of Interest needs several competences, which can be for example formalized through disciplinary modules. Therefore, the lower layer *Competences* is part of the MBSE Architectural Framework.

As mentioned before, the ultimate aim of the MBSE Architectural Framework is the development of Systems of Interest. However, the same framework can be employed for the development of multiple systems operating together according to the concept of *System of Systems*, which defines the fourth layer, the highest one.

The present deliverable focuses on the *Design System* layer, although the proposed guideline is used to represent the architectures of a *System of Interest*.

Several architectural frameworks are available in literature (e.g. Zachman's Framework [8], DoDAF [9], MODAF [10], NAF [11] and TOGAF [12]). They are all characterized by the following elements:

- Ontology: definition of the concepts composing the architecture and their relationships [13].

- Viewpoint: convention for the construction, interpretation and use of architectures from the perspective of specific system concerns [7].
- Process: logical sequence of tasks performed to achieve a particular objective [14].

As explained in Section 2.1, this deliverable focuses on the third and fourth steps of the Systems Engineering approach being adopted in the AGILE 4.0 project, namely *System Architecting* and *System Synthesis*. Therefore, the ontology presented in this deliverable is about the elements addressed during system architecting activities, e.g. *function, logical/physical component, architecture design space*. Analogously, the viewpoints presented in the deliverable are focusing on the *System Architecture* perspective. Finally, the system architecting process developed in WP2 includes the activities of: identification of system functions, allocation of function to logical components, characterization of logical components, generation of instances of system architectures and determination of physical components of generated system architectures. The ontology, viewpoints and process for system architecting are described in the following Chapters.

It should be finally noted that the AGILE 4.0 MBSE architectural framework is employed in the frame of the present deliverable to address the definition of Systems Of Interest. However, the proposed MBSE architectural framework can also be adopted to address all the layers depicted in Fig. 3, therefore targeting also the development of the Design System.

4 ONTOLOGY

4.1 Introduction

Three types of *System Architectures* are addressed in the present deliverables: functional, logical and physical architectures [15]. Both, document and model-based approaches, can be adopted to represent these three types. The following subsections define some elements regarding each kind of system architecture.

4.1.1 Functional architecture

The aim of the *Functional Architecting* part of the system architecting process is to identify all the functions that the system (through its components) should fulfill [16]. These functions are called *boundary functions* [17], and they are determined during the architecting process that focuses on a specific *level of elaboration of the system* under design.

The definitions of the main concepts addressed in the Functional Architecting part of the system architecting process are reported below together with some examples:

- **System of Interest** (or **System under design**): product being developed by the design team [18]
- **Level of elaboration (of the system)**: specific level of the System of Interest in a hierarchy ranging from a top system-level (e.g. aircraft-level) to a lower subsystem-level (e.g. wing-level) to a bottom component-level (e.g. spar-level)

> **Example 1**
>
> **System of Interest:** Aircraft
>
> **1st level of elaboration:** Aircraft
> **2nd level of elaboration:** Fuselage, Wing, **Fuel system**, Engine, Tail Plane
> **3rd level of elaboration:** Tanks, pumps, tubes

- **Architecture (of a level of elaboration)**: describing how a system's level of elaboration is organized and operates, by allocating functions (what the system does) to components (how the system performs its functions), and the definition of relationships among the components [19]

> **Example 2**
>
> **1st level of elaboration → 2nd level of elaboration:**
> **Aircraft** → Fuselage, Wing, On-board system, Engine, Tail Plane
>
> **2nd level of elaboration → 3rd level of elaboration:**
> **Fuel system** → Tanks, pumps, tubes

- **Actor**: person (human) or system, including a stakeholder, who is operating with the system's level of elaboration [18].

> **Example 3**
>
> **Level of elaboration:** Aircraft
> **Actor:** Pilot, passenger, Ground Power Unit, Maintainer, Air Traffic Control
>
> **Level of elaboration:** Fuel System
> **Boundary functions:** Engine, Electric System, Refueling operator

- **Boundary function**: *"what"* a level of elaboration of the system should provide through its architecture, determined from functional requirements. Boundary functions are solution-neutral, and they are not determined on the basis of which components are part of the system architecture [17].

Example 4

Level of elaboration: Aircraft
Boundary functions: Generate lift, Provide thrust, Contain payload, Accommodate payload, Control environment, Control the aircraft, Prevent icing

Level of elaboration: Fuel System
Boundary functions: Store fuel, Pressurize fuel, Distribute fuel, Jettison fuel, Measure fuel quantity, Transfer fuel

- **Use Case**: Single high-level function of the level of elaboration assessed in a specific condition. This condition refers to a particular situation during/in which the system operates (e.g. in cruise, in emergency, during landing, at cruise altitude, on ground, in 1-engine-out condition). At least a Use Case refines a functional requirement. A Use Case contains one or more boundary functions that are derived from the high-level function [18]. A high-level function is a generic function that belongs to the elaboration level (for example, an aircraft has to *fly*). Derived boundary functions specifies the high-level function (for example, in order to fly, components of the aircraft shall *generate lift* and *generate propulsive power*)

Example 5

Top Function: Control the aircraft
Use Case 1: Control the aircraft **in flight**
Use Case 2: Control the aircraft **on ground**

Top Function: Provide fuel
Use Case 1: Provide fuel **in nominal cruise**
Use Case 2: Provide fuel **in 1-engine-out condition**

- **Functional scenario**: representation in time of a Use Case, through all the boundary functions included in it (adapted from [20]).

4.1.2 Logical Architecture

The aim of the *Logical Architecting* part of the system architecting process is to generate multiple architectures having different *logical components* fulfilling all the boundary functions defined in the previous system architecting part [18]. Components can also induce other functions – which are therefore called *induced functions* – that are fulfilled by other components [17]. Moreover, each component can be specified according to some *characterization options* (i.e. number of instances, QoIs with possible values and types of component) and *connection options* (*ports* define interfaces between components). The *Architectural Design Space* represents all the possible system architectures that can be generated by taking *architectural design decisions* [19]. These decisions encompass the selection of components and the specification of characterization and connection options. These functions are called boundary functions, and they are determined during the architecting process that focuses on a specific level of elaboration of the system under design.

The definitions of the main concepts addressed in the Logical Architecting part of the system architecting process are reported below together with some examples:

- **(Logical) component**: element implemented into the system's level of elaboration and fulfilling a function (boundary or induced) (adapted from [19])

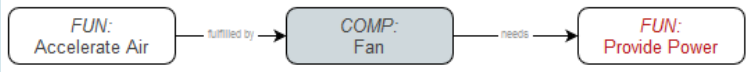
Example 7

Level of elaboration: Aircraft
Components (*functions*): Wing (*F: provide lift*), Engine (*F: provide thrust*), Fuselage (*F: contain payload*), Landing gear (*F: perform ground maneuvers*)

Level of elaboration: Fuel System
Components (*functions*): Fuel tank (*F: store fuel*), Fuel pump (*F: pressurize fuel*), Fuel line (*F: distribute fuel*)

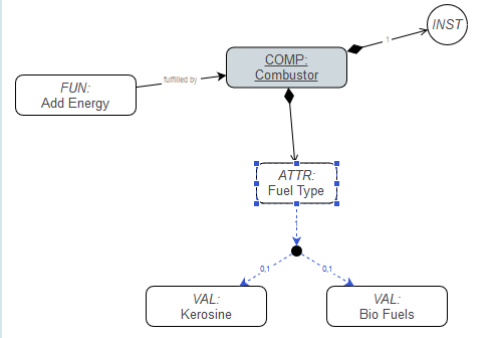
- **Induced function:** function that is induced from the selection of a component. These functions are *required* by a component in order to perform the functions it fulfills [17].

Example 8



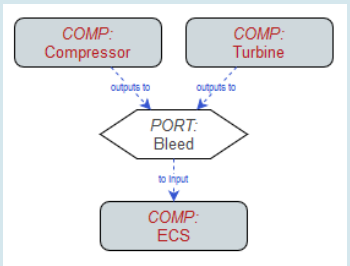
- **Characterization option (of a component):** number of component instances and attributes of a component (e.g. "conventional" vs. "strut-braced" wing) [21]

Example 9



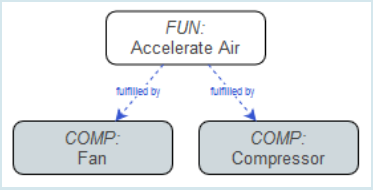
- **Connection option (of a component):** connections between components, for example representing location of the component with respect to other components or within the whole system architecture (e.g. "wing-mounted" vs. "tail-mounted" engines), or flows (e.g. mass, information, energy, etc.) [21]

Example 10



- **Architectural decision**: selection of logic components for fulfilling functions, selection of component attribute values, decision on the number of components, decision on connections between components [19]

Example 10

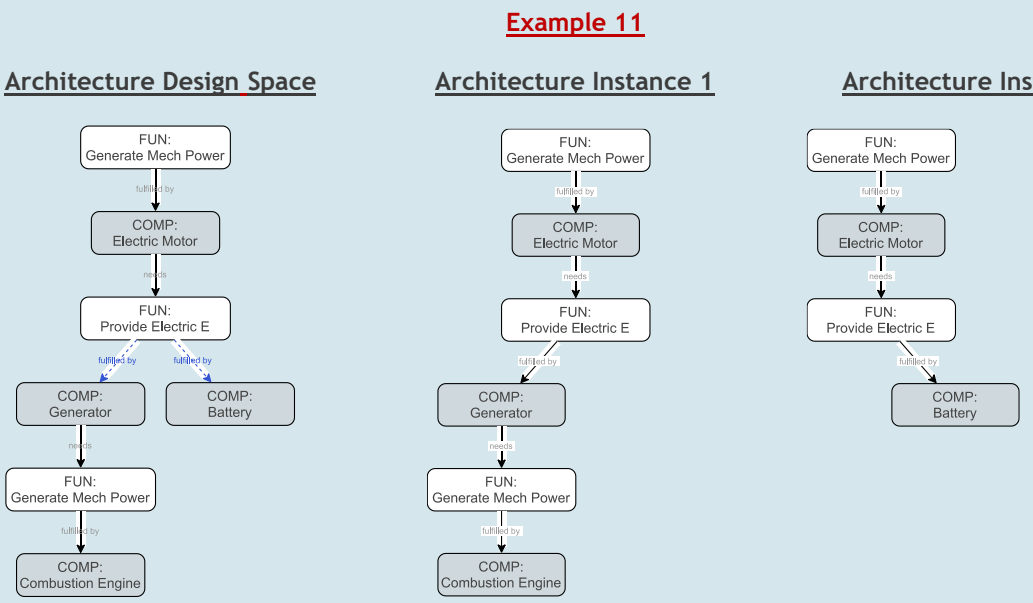


(the function ***Accelerate Air*** is fulfilled by EITHER a **Fan** OR a **Compressor**)

- **Architectural Design Space**: model representing all possible functions, components and component characterization and connection decisions [21].

- **Architecture instance**: specific architecture instantiated from the Architectural Design Space by assigning values to architectural decisions. Additionally, an architecture instance represents a real configuration of a system and provides the logical basis for a physical architecture that can be simulated/evaluated [21].

Example 11



4.1.3 Physical Architecture

The main aim of the *Physical Architecting* part of the system architecting process is to instantiate the logical architectures generated in the previous step by identifying *physical components* [15]. In AGILE 4.0, physical components are characterized by *Quantities of Interest (QoI)* (e.g. masses, performance characteristics, dimensions), which can be determined through MDAO processes. In fact, this step represents the shift from *components* perspective to *disciplinary competence* perspective. In addition, results of MDAO processes are verified against the non-functional requirements identified during the Application Case Definition part of the Systems Engineering Technical Process, i.e. the first two steps in Fig. 1.

The definitions of the main concepts addressed in the Physical Architecting part of the system architecting process are reported below together with some examples:

- **Quantity of Interest (QoI):** value addressed in MDAO processes. From the point of view of the architecture evaluation, a QoI represents either an input to or an output from the evaluation. From the point of view of the MDAO process, a QoI can be interpreted as a design variable (input), a configuration value (input), an objective (output), a constraint (output) or any other output parameter that can be monitored during a MDAO process.

- **(Physical) component:** instance of a logical component through the definition of QoIs.

> **Example 12**
>
> **Logical Component:** Engine
> **Physical Component:** CFM56-5B
> **Quantities of Interest:** Dry mass: 2500 kg; Takeoff thrust: 140 kN, By-pass Ratio: 6.0; Specific Fuel Consumption (@Cruise): 0.545 lb/(lbf·h)

- **Requirements Verification:** process addressed to prove that the obtained solution (e.g. physical architecture) complies with what stated by (non-functional requirements).

> **Example 13**
>
> **Physical Component:** CFM56-5B (engine)
> **Quantities of Interest:** Dry mass: 2500 kg; Takeoff thrust: 140 kN
>
> **Requirement 1:** The engine shall have a dry mass of max 2450 kg → Non-compliant
> **Requirement 2:** The engine shall generate a thrust of min 135 kN in takeoff → Compliant

4.2 Ontology of System Architecture

An ontology representing the main concepts of system architectures (e.g. functions, components, architectural space) and their relations (e.g. function induced by a component) is described in the present section and depicted in Fig. 4. The model of the system architecting ontology can be visualized through a *SysML Internal Block Diagram*, where all the system architecting elements are represented by the new stereotype *«ontology concept»*, introduced in the project.

The first relevant concept is the ***Level of the System of Interest***, which is the level of elaboration of the system that has to be architected. This level of the system has to provide different ***high-level functions***, each one of them expressed by a ***functional requirement***. These types of requirements are refined by ***use cases***, which describe how ***actors*** interact with the level of abstraction of the system *(e.g. a pilot provides commands to the aircraft)*. In addition, use cases are made of multiple ***boundary functions***, which are specializations of ***functions***. In fact, other types of functions are determined in the system architecting process, i.e. the ***induced functions***. The system functional behavior described by each use case can be represented in time through ***functional scenarios***.

Both boundary and induced functions are defining the ***system functional architectural design space***, in which ***solution-neutral and solution-specific functional architectures*** are included. The former ones are made of only boundary functions, while the latter ones also include induced functions. Both functions should be allocated to ***logical components***, which all together form the ***system logical architectural space***. Multiple characterization and connection options characterize the different system components. ***Architecting decisions*** are also part of the system architectural space, and they are taken by the system architects in order to define specific ***system logical architectures***.

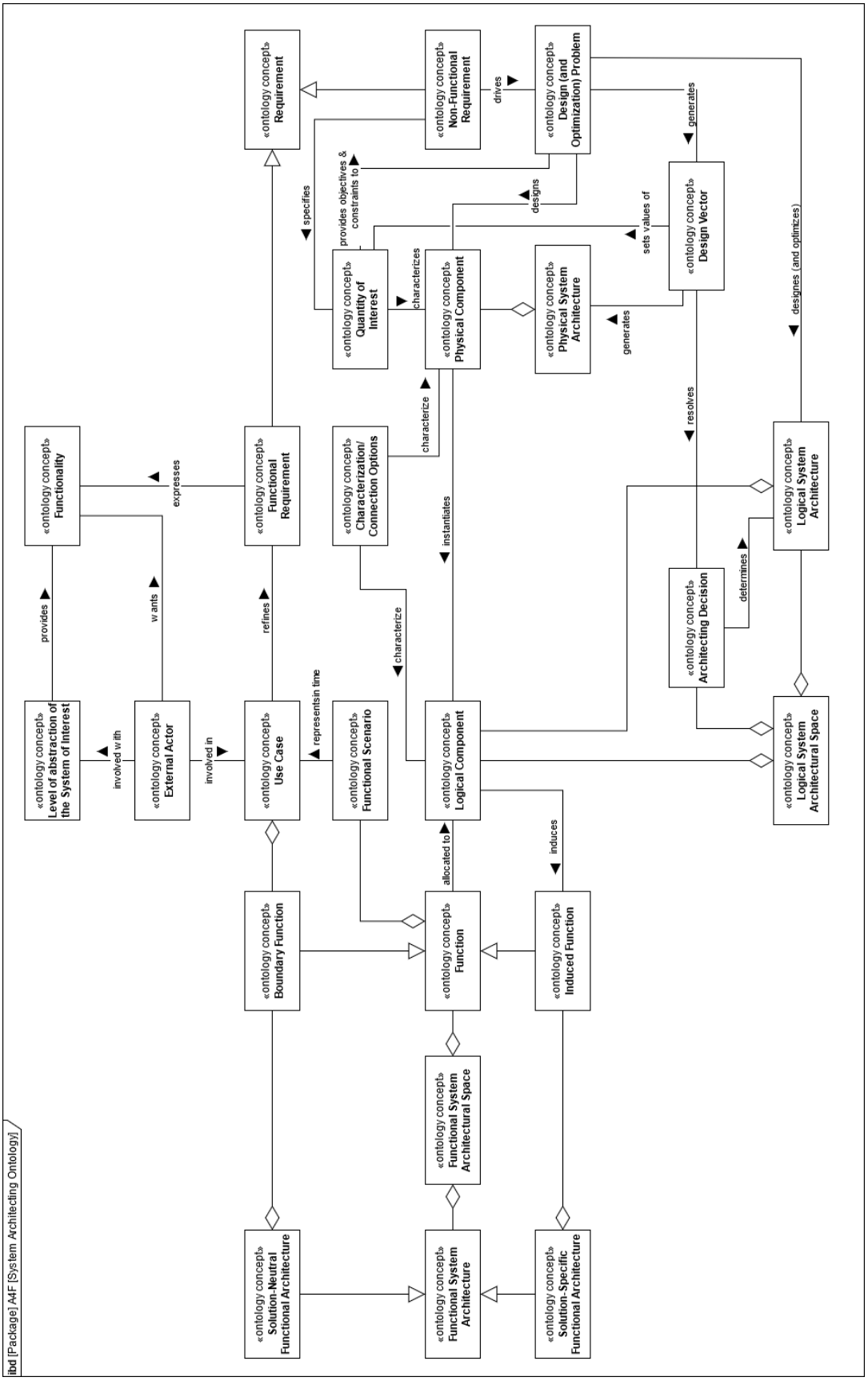


Fig. 4: AGILE 4.0 project ontology of System Architecture represented through a SysML Internal Block Diagram.

Each logical architecture is then ***designed and optimized*** through an MDAO problem, which is driven by ***QoIs*** (e.g. constraints) extracted from the ***non-functional requirements*** and included in ***design vectors***. The results that are derived from the execution of MDAO problems determine and/or size ***physical components***, which are instantiations of logical components. All the physical components together form the system ***physical architectural space***, which includes multiple ***system physical architectures***.

5 VIEWPOINTS

Multiple viewpoints (see the definition in Chapter 3) are part of the MBSE architectural framework for the representation of the system architecture model. All the viewpoints are collected in the SysML Package Diagram of Fig. 5

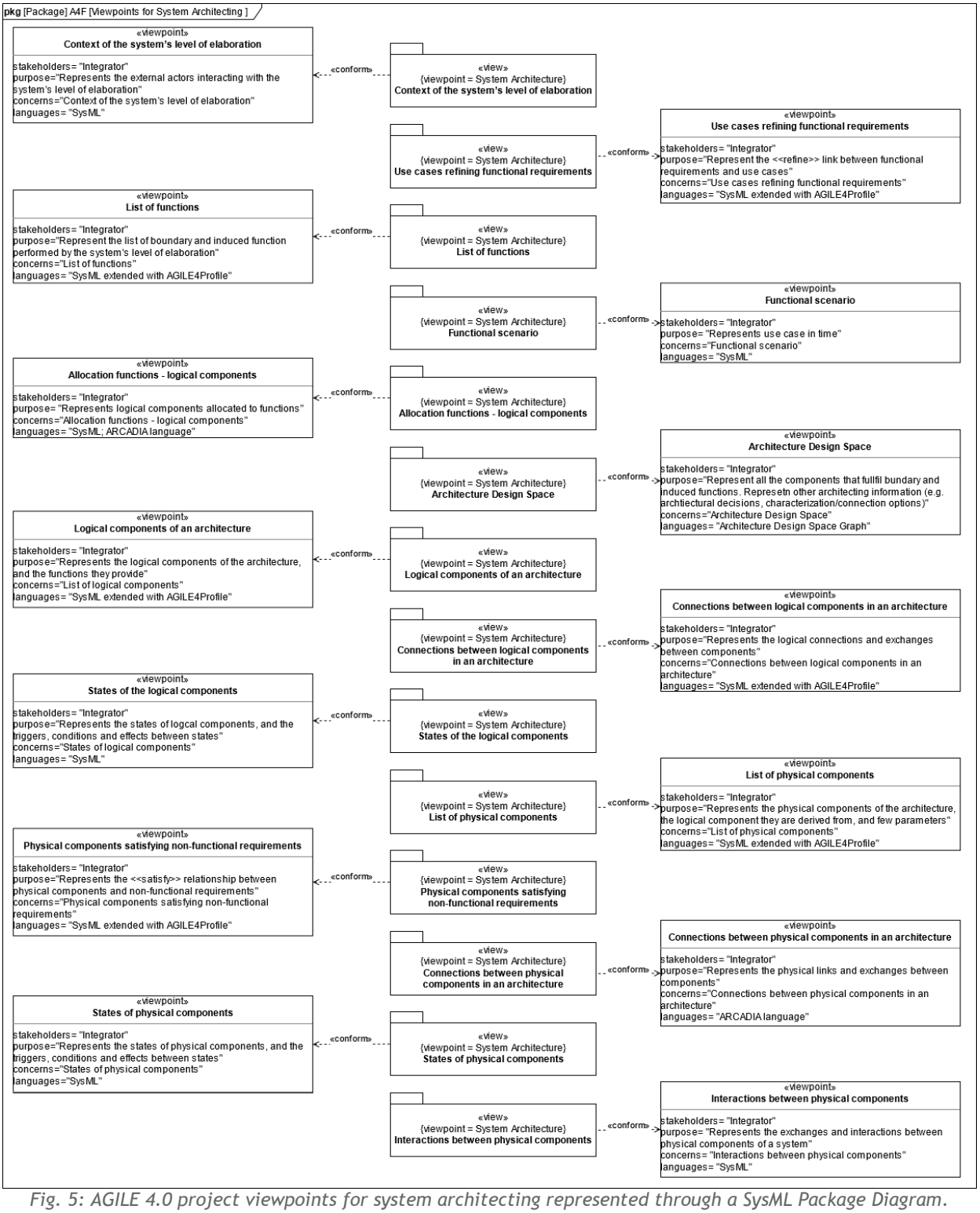


Fig. 5: AGILE 4.0 project viewpoints for system architecting represented through a SysML Package Diagram.

As described in the following subsections, the proposed viewpoints make use of different modeling languages. The most used language is an extension of SysML, named AGILE4Profile. As suggested by its name, this profile has been developed within the frame of the AGILE 4.0 project, and it extends the standard SysML by introducing new stereotypes, as described in the Appendix. The modeling language of the MBSE process ARCADIA [22] (implemented in the tool Capella) is used as well in some viewpoints. Finally, the Architecture Design Space Graph (ADSG) [21] is employed as a language to model architectural design spaces. Deliverable D4.6 [6] reports more details about how the various modeling languages are employed in AGILE 4.0 to represent system architectures.

5.1 Functional Architecture Viewpoints

During the system functional architecting process, four views can be created by adhering to the viewpoints listed in the present subsection.

Context of the system's level of elaboration Viewpoint

AIM
This Viewpoint is used to represent the actors interacting with the system's level of elaboration
CONCERNS ADDRESSED
Context of the system's level of elaboration
REPRESENTATION
SysML Use Case Diagram (see D4.6 [6])

Use cases refining functional requirements Viewpoint

AIM
This Viewpoint is used to represent the «refine» link between functional requirements and use cases
CONCERNS ADDRESSED
Use cases refining functional requirements
REPRESENTATION
SysML Requirements Diagram (see D4.6 [6])

List of functions Viewpoint

AIM
This Viewpoint is used to represent the list of boundary and induced functions performed by the system's level of elaboration
CONCERNS ADDRESSED
List of functions
REPRESENTATION
SysML Block Definition Diagram (see D4.6 [6])

Functional scenario Viewpoint

AIM
This Viewpoint is used to represent use case in time
CONCERNS ADDRESSED
Functional scenario
REPRESENTATION
SysML Activity and Sequence Diagrams (see D4.6 [6])

5.2 Logical Architecture Viewpoints

Five viewpoints can be used to represent functional architectures.

Allocation functions - logical components Viewpoint

AIM
This Viewpoint is used to represent which logical components are allocated to which system functions
CONCERNS ADDRESSED
Allocation functions – logical components
REPRESENTATION
SysML Activity Diagram; ARCADIA language ([LAB diagram]) (see D4.6 [6])

Architecture Design Space Viewpoint

AIM
This Viewpoint is used to represent all the components that fulfill the boundary and induced functions and other architecting information (e.g. architectural decisions, characterization/connection options
CONCERNS ADDRESSED
Architecture Design Space
REPRESENTATION
Architecture Design Space Graph (see D4.6 [6])

Logical components of an architecture Viewpoint

AIM
This Viewpoint is used to represent the logical components of the architecture, and functions they provide
CONCERNS ADDRESSED
List of logical components
REPRESENTATION
SysML Block Definition Diagram (see D4.6 [6])

Connections between logical components in an architecture Viewpoint

AIM
This Viewpoint is used to represent the connections and exchanges between logical components
CONCERNS ADDRESSED
Connections between logical components
REPRESENTATION
SysML Internal Block Diagram (see D4.6 [6])

States of the logical components Viewpoint

AIM
This Viewpoint is used to represent the states of logical components, and the triggers, conditions and effects associated to changes from state to state
CONCERNS ADDRESSED
States of the logical components
REPRESENTATION
SysML State Machine Diagram (see D4.6 [6])

5.3 Physical Architecture Viewpoints

Five viewpoints are recommended by the AGILE 4.0 project consortium to represent physical architectures.

List of physical components Viewpoint

AIM
This Viewpoint is used to represent the physical components of the architecture, the logical components they are derived from, and some parameters (QoIs)
CONCERNS ADDRESSED
List of physical components
REPRESENTATION
SysML Block Definition Diagram (see D4.6 [6])

Physical components satisfying non-functional requirements Viewpoint

AIM
This Viewpoint is used to represent the «satisfy» relationship between physical components and non-functional requirements
CONCERNS ADDRESSED
Physical components satisfying non-functional requirements
REPRESENTATION
SysML Requirement Diagram (see D4.6 [6])

Connection between physical components in an architecture Viewpoint

AIM
This Viewpoint is used to represent the physical links and exchanges between components
CONCERNS ADDRESSED
Connection between physical components in an architecture
REPRESENTATION
ARCADIA language ([PAB] diagram) (see D4.6 [6])

States of physical components Viewpoint

AIM
This Viewpoint is used to represent the represent the states of physical components, and the triggers, conditions and effects associated to changes from state to state
CONCERNS ADDRESSED
States of physical components
REPRESENTATION
SysML State Machine Diagram (see D4.6 [6])

Interactions between physical components Viewpoint

AIM
This Viewpoint is used to represent the exchanges and interactions between physical components of a system
CONCERNS ADDRESSED
Interactions between physical components
REPRESENTATION
SysML Sequence Diagram (see D4.6 [6])

6 PROCESS

This Chapter presents the process developed in AGILE 4.0 for the architecting of complex systems. As explained before, three types of architectures are considered: functional, logical and physical. Therefore, the system architecting process defined in the project can be divided into three parts, as described in the following subsections.

6.1 *Functional* system architecting process

The functional architecting part of the system architecting process is schematized in Fig. 6, and its aim is to identify all the functions that the system (through its components) should fulfill.

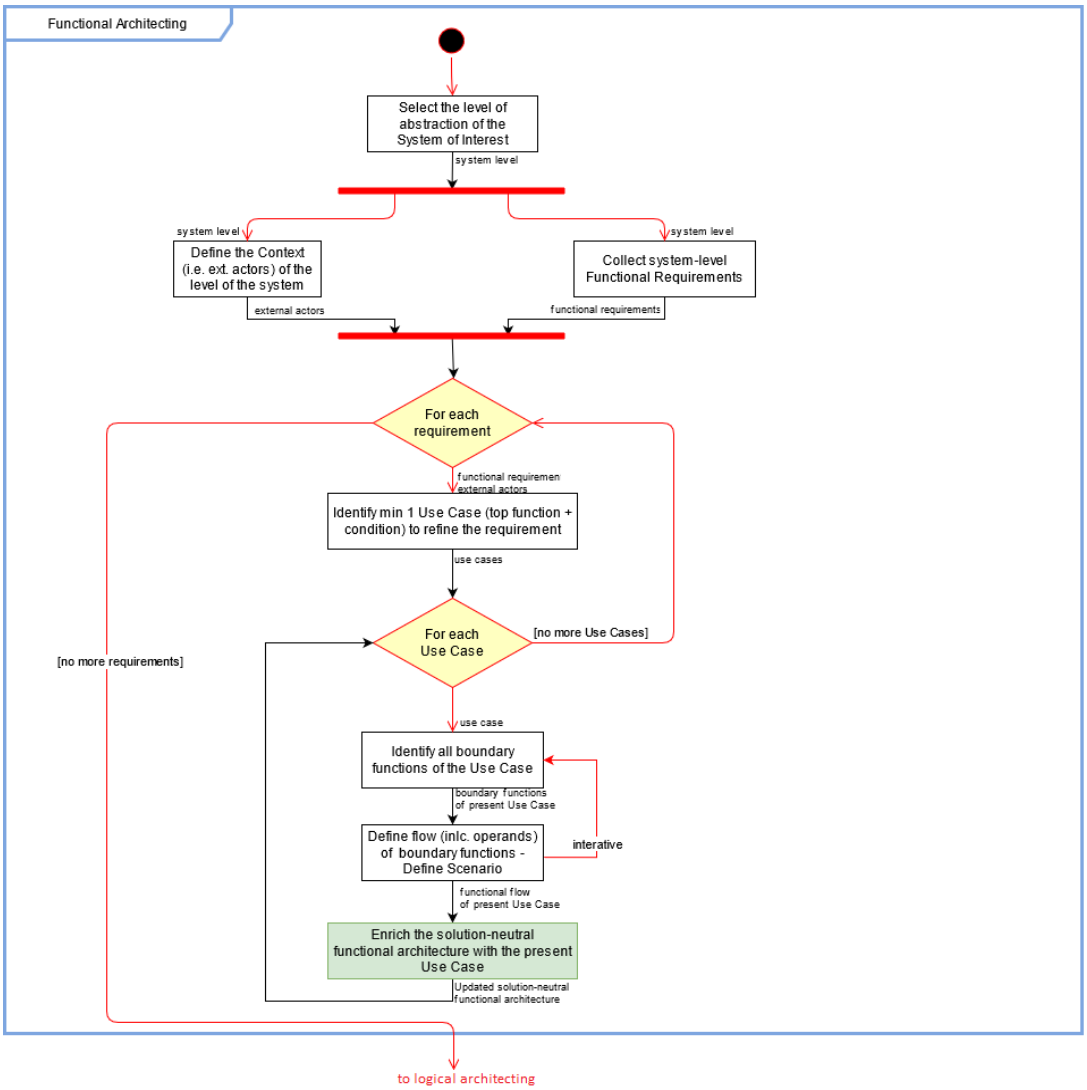


Fig. 6: Functional part of the AGILE 4.0 system architecting process.

The functional architecting process focuses on as specific system's level of abstraction (e.g. aircraft-level or wing-level), for which all the functional requirements are collected. Boundary functions are then derived by refining requirements through use cases. At least a use case refines a functional requirement, as recommended by several methodologies available in literature, e.g. FAR (Functional Architecture by use case Realizations) [20], FAS (Functional Architectures for Systems) [23] and SYSMOD [18]. A use case contains one or more boundary functions that are derived from the high-level function. A use case – including all the boundary functions that belong to it – can be represented in time, entailing therefore a functional scenario. Once all

the boundary functions that the specific level of elaboration of the system has to perform are identified, the architecting process can proceed with the following part, regarding the logical architecting.

6.2 *Logical* system architecting process

The logical architecting part of the system architecting process is schematized in Fig. 7, and its aim is to generate multiple architectures characterized by different logical components fulfilling all the boundary functions defined in the previous part.

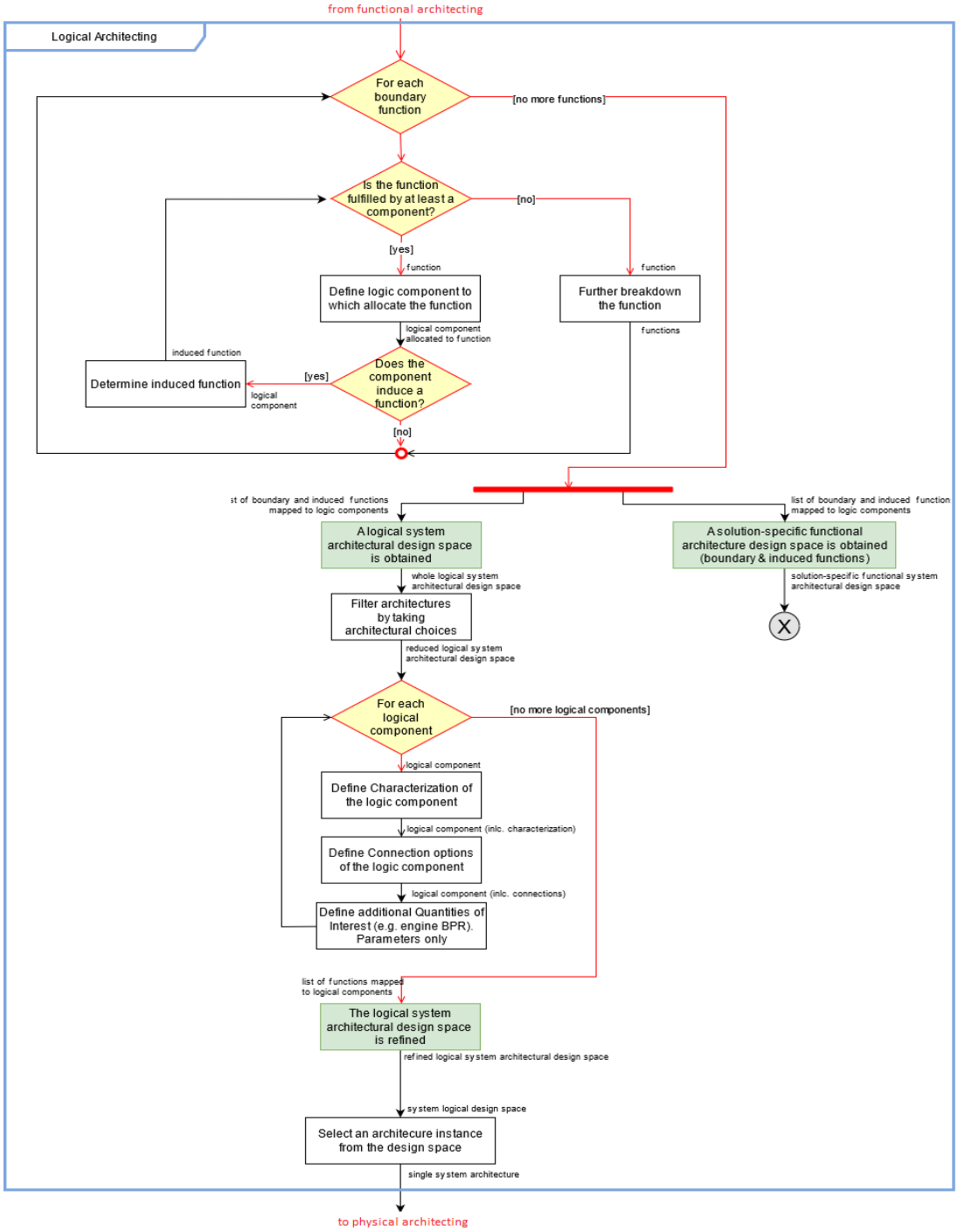


Fig. 7: Logical part of the AGILE 4.0 system architecting process.

Each boundary function previously defined is allocated to a least a logical component. For instance, the components *turbofan engine* or *turboprop engine* can fulfil the function *provide propulsive power*. Components can also induce other functions – the so-called *induced functions* – that are fulfilled by other components. For example, an engine component would induce the induced function *provide fuel*. Moreover, each component can be specified according to some characterization options (i.e. number of instances, QoIs with possible values and types of component) and connection options (ports define interfaces between components). The Architectural Design Space represents all the possible system architectures that can be generated by taking architectural design decisions. These decisions encompass the selection of components, and the specification of characterization and connection options. The architecting decisions allow the selection of one or more architecture instances from the Architectural Design Space that can be addressed in the physical system architecting process.

6.3 *Physical* system architecting process

The aim of the physical architecting part of the system architecting process (see Fig. 8) is to identify suitable physical components compliant with the non-functional requirements of the system's level of elaboration. A physical component instantiates a logical one by specifying characteristics such as performance, dimension, or material attributes. For example, the *CFM56* is a physical component characterized by certain mass, thrust, fuel consumption, which instantiates the logical component *turbofan engine*. Physical architectures can be derived from logical ones through the formulation and execution of MDAO processes. An MDAO process can be set-up by identifying the test cases (e.g. disciplinary competence, as aerodynamics and structure) required to verify the non-functional requirements of the system's level of elaboration [24]. Moreover, the mapping between architecture components and disciplinary competences can be done at this stage, as recommended in [25], in order to make sure that the design team has all the necessary disciplinary tools required to design and optimize a specific logical system architecture. Another example of linking the system architecting process with the MDAO execution and the physical architecture is explored in [26]. Once an MDAO process has been formulated and (successfully) executed, system solutions are determined. This step entails the verification of requirements at the system's level of elaboration and the determination of a physical architecture, designed and possible optimized.

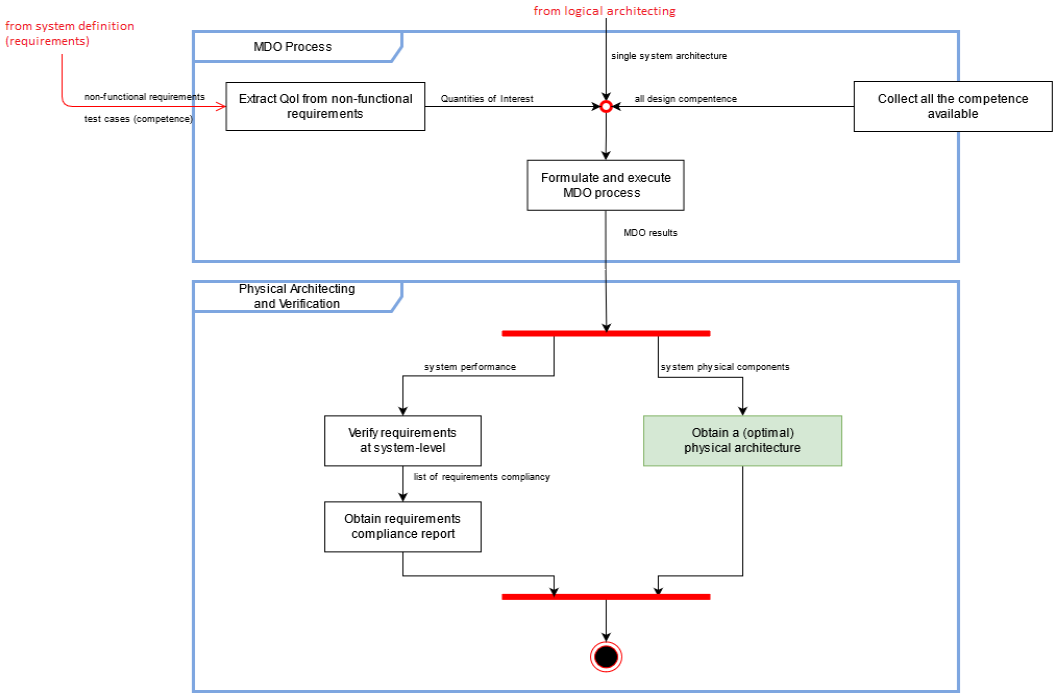


Fig. 8: Physical part of the AGILE 4.0 system architecting process.

7 CONCLUSIONS AND FUTURE DIRECTIVES

This deliverable presents the process and guideline for the system architecting activities defined in AGILE 4.0, meaning the identification of functional, logical and physical architectures. The process, ontology and viewpoints for system architecting are part of the MBSE architectural framework developed in the project. The process and guideline addressed in the deliverable can be applied in both document and model-based approaches.

The process, ontology and viewpoints for the system architecting have been iteratively developed in close collaboration with the envisaged stakeholders: the Application Case developers in AGILE 4.0 WPs 6, 7 and 8. During the development of the process and guideline, it became clear that awareness and close involvement of stakeholders were important elements for acceptance, validation and are an essential basis for successful application of the method.

Concluding, it can be stated that the process and guidelines developed in AGILE 4.0 and described in the present deliverable (together with the enabling tools realized in the project) can effectively support and improve the system architecting activities. However, some main limitations have also been identified, and additional research should be carried out beyond the AGILE 4.0 project.

The first limitation is related to the iterative aspect of the system development process. The system architecting process defines a new and lower level of elaboration, since components are identified to fulfill all the (boundary) functions entailed from the high-level function belonging to an upper level (e.g. aircraft-level). These components and their functions should generate new functional and non-functional requirements at the new level of elaboration, therefore starting again the system architecting process. Nevertheless, this “re-iteration” of the process has not be fully addressed in the project.

The second limitation focuses on the transition from logical to physical architecting. In reality, this transition is not straightforward, but it involves several iterations, until the physical architecture is complete. This iterative transition is not fully considered in the physical architecture process and the associated viewpoints developed in AGILE 4.0 (and therefore neither completely supported by the enabling tools realized in the project).

REFERENCES

[1] AGILE 4.0, "D1.3 Process definition A4F for use case definition," AGILE 4.0 project (H2020-815122), 2020.

[2] AGILE 4.0, "D1.2 Requirements & Specifications for AGILE 4.0 enabling technologies," AGILE 4.0 project (H2020-815122), 2020.

[3] AGILE 4.0, "D6.4 Production driven stream use cases MDO framework implementation," AGILE 4.0 project (H2020-815122), 2021.

[4] AGILE 4.0, "D7.4 Certification driven stream use cases MDO framework implementation," AGILE 4.0 project (H2020-815122), 2021.

[5] AGILE 4.0, "D8.4 Upgrade driven stream use cases MDO framework implementation," AGILE 4.0 project (H2020-815122), 2021.

[6] AGILE 4.0, "D4.6 Models for architecture definition," AGILE 4.0 project (H2020-815122), 2021.

[7] International Organization for Standardization, "ISO/IEC/IEEE 42010 - Systems and software engineering - Architecture description," 2011.

[8] J. A. Zachman, "A framework for information systems architecture," *IBM systems journal*, vol. 26, no. 3, pp. 276-292, 1987.

[9] U.S. DoD, "The DoDAF Architecture Framework Version 2.02," 2010. [Online]. Available: https://dodcio.defense.gov/Library/DoD-Architecture-Framework/.

[10] UK Ministry of Defence, "MOD Architecture Framework," 2012. [Online]. Available: https://www.gov.uk/guidance/mod-architecture-framework.

[11] NATO, "NATO Architecture Framework Version 4," 2018. [Online]. Available: https://www.nato.int/cps/en/natohq/topics_157575.htm. [Accessed 11 11 2022].

[12] The Open Group, "The TOGAF® Standard, Version 9.2," 2018. [Online]. Available: https://www.opengroup.org/togaf.

[13] J. Holt and S. Perry, SysML for systems engineering, London: IET, 2008.

[14] J. A. Estefan, "Survey of Model-Based Systems Engineering (MBSE) Methodologies," Incose MBSE Focus Group, 25(8), 1-12, 2008.

[15] A. Levis, "System Architectures," in *Handbook of Systems Engineering and Managemen*, New York, John Wiley & Sons, 1999, pp. 427-454.

[16] J. G. Lamm and T. Weilkiens, "Tag des Systems Engineering," in *Gesellschaft für Systems Engineering e.V.*, Munich (DE), 2010.

[17] D. Mavris, C. de Tenorio and M. Armstrong, "Methodology for Aircraft System Architecture Definition," in *AIAA ScieTech*, Reno (US-NV), 2008.

[18] T. Weilkiens, J. G. Lamm, S. Roth and M. Walker, Model-based system architecture, John Wiley & Sons, 2015.

[19] E. Crawley, B. Cameron and D. Selva, System Architecture. Strategy and Product Development for Complex Systems, Harlow (UK): Person Education Limited, 2016.

[20] M. Eriksson, K. Borg and J. Börstler, "Use cases for systems engineering—an approach and empirical evaluation," *Systems Engineering*, vol. 11, no. 1, pp. 39-60, 2008.

[21] J. H. Bussemaker, P. D. Ciampa and B. Nagel, "System Architecture Design Space Exploration: An Approach to Modeling and Optimization," in *AIAA AVIATION 2020 FORUM*, Virtual Event, 2020.

[22] P. Roques, "MBSE with the ARCADIA Method and the Capella Tool," in *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, 2016.

[23] J. G. Lamm and T. Weilkiens, "Method for deriving functional architectures from use cases," *Systems Engineering*, vol. 17, no. 2, pp. 225-236, 2014.

[24] A. Bruggeman, T. van der Laan, T. van den Berg, B. van Manen and G. La Rocca, "An MBSE-Based Requirement Verification Framework to support the MDAO Design Process," in *AIAA Aviation Forum*, Chicago (US-IL), 2022.

[25] J. H. Bussemaker, P. Ciampa and B. Nagel, "From System Architecting to System Design and Optimization: A Link Between MBSE and MDAO," in *INCOSE Symposium*, Detroit (USA-MI), 2022.

[26] A. K. Jeyaraj, N. Tabesh and S. Liscouet-Hanke, "Connecting Model-based Systems Engineering and Multidisciplinary Design Analysis and Optimization for Aircraft Systems Architecting," in *AIAA Aviation FORUM*, Virtual, 2021.