AIAA AVIATION Forum
June 27-July 1, 2022, Chicago, IL & Virtual
AIAA AVIATION 2022 Forum

10.2514/6.2022-3720

An MBSE Architectural Framework for the Agile Definition of Complex System Architectures

Luca Boggero*, Pier Davide Ciampa†, Björn Nagel‡
German Aerospace Center (DLR), Institute of System Architectures in Aeronautics, Hamburg, Germany

In the recent years, a shift from document-based to model-based approaches is going on within many organizations and industries involved in the development of complex systems. Model Based Systems Engineering (MBSE) methods and tools are in fact gaining more and more popularity due to all their claimed benefits over traditional document-based approaches, including for instance enhanced design quality of systems, clearer development of system requirements and specifications and improved communications within the design teams. However, these benefits can be possible only if recommendations on how generating and representing design information during the development process are made available. The present paper introduces a new model-based *architectural framework*, i.e. a guideline that leverages a modeling approach for the development and representation of complex systems. More specifically, the MBSE architectural framework addressed in this paper focuses on the system architecting activities of a Systems Engineering Product Development process, i.e. when multiple conventional and innovative solutions of the systems are generated to address all the system stakeholder expectations. The proposed architectural framework aims at fostering the *agility* of the development of complex systems, in order to streamline, improve and accelerate their architectures definition and modeling through an MBSE approach. The paper provides details of the MBSE architectural framework, including the means to produce and represent all the system development information.

Nomenclature

ADSG	=	Architecture Design Space Graph
INCOSE	=	International Council on Systems Engineering
MBSE	=	Model Based Systems Engineering
MDO	=	Multidisciplinary Design Optimization
SysML	=	System Modeling Language

I. Introduction

NEW challenges as climate neutrality and the continuously increasing demand of higher performance are facing the aviation sector for the next decades. All these challenges are pushing aeronautical industries and research centers to develop radical innovative concepts characterized by novel and disruptive technologies, including for example innovative propulsion systems (e.g. hybrid-electric and hydrogen-powered systems), more efficient high aspect ratio laminar wings, more electric on-board systems, new materials.

The introduction of such radical innovative technologies makes the aeronautical system even more complex as ever. The complexity of the development process of aeronautical systems is reflected in an increased number of designers and a larger amount of data and information produced. Design data regard multiple aspects of the aeronautical product under development. For example, these data include requirements, specifications, descriptions and interfaces of the several systems, components and parts of the aircraft. Moreover, data and information include

* Research Scientist, Institute of System Architectures in Aeronautics, Aircraft Design & System Integration, Hamburg, Luca.Boggero@dlr.de.
† Head of MDO Group, Institute of System Architectures in Aeronautics, Aircraft Design & System Integration, Hamburg, Pier.Ciampa@dlr.de, AIAA MDO TC member.
‡ Founding director, Institute of System Architectures in Aeronautics, Hamburg, Bjoern.Nagel@dlr.de.

organizational aspects of the entire development process, like design decisions, project life-cycle, which designers are involved. All the different elements composing the large amount of design data are related together. For instance, different solutions are derived according to different design decisions, which are taken on to satisfy the system requirements. In case some specific elements are modified, impacts on the rest of the design data and information are originated. In addition, data and information are authored by multiple engineers and designers from different departments or organizations of the aeronautical supply chain. Therefore, many relationships among all the elements might be hidden, therefore making difficult or even impossible the traceability among all these elements.

A. Shifting from a document-based to a model-based architectures definition

An *architectural framework* can be a solution supporting the organization of all the data produced during the development process. 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 stakeholder***s" [1]. **The *architecture* in this definition represents the structure and behavior of a *system***, where a ***system*** is a "***set of entities and their relationships, whose functionality is greater than the sum of the individual entities***" [2]. For example, an aircraft is a *system*, since it can fly and transport payload (*system functionality*) thanks to all its *entities* joined together, as engines, fuselage, wings and on-board systems.

Therefore, an *architecture* describes different aspects of the *system.* An *architecture* can represent for instance all the parts that compose the *system*, its life-cycle, how it is operated by external users and a lot of other information. A *system architecture* instead is a specific *architecture*, which represents only all the *components* (*entities* in case of *system architectures*) that are part of the system and how they are connected together. An *architectural framework* provides the guidelines for the standard representation of the multiple *system architectures.* Several *architectural frameworks* are available in literature, e.g. Zachman's Framework [3], DoDAF [4], MODAF [5], NAF [6] and TOGAF [7].

Traditionally, designers follow the guidelines recommended by *architectural frameworks* for the preparation of documents addressing the different aspects of the *system*. Textual documents or tables collect system requirements, specifications, technical descriptions, but they also track the design decisions taken, showing the evolution of the *system* from the initial concept to its realization. Such document-based approach is however negatively characterized by limitations and disadvantages, which include for example poor traceability, ambiguity, misunderstandings, lack of clarity. Therefore, novel approaches based on models are spreading among industries and organizations due to their potential benefits in terms of easier design activities, enhanced design quality, better system specification and improved communications within the design team [8]. Models can indeed be built and employed in place of documents to clearly represent *architectures*. Therefore, all these motivations have given great popularity in the last decade to *Model Based Systems Engineering* (MBSE), where all the activities of a *Systems Engineering Product Development* process (such as definition of customer needs, identification of system functionalities, collection of requirements, generation of alternative system architectures and verification and validation tasks) are supported by models instead of documents. Due to the exponential rising in *systems* complexity caused by new demands in higher performance, lower environmental impact and augment of functionalities, MBSE is expected to play an increasing role in the field of Systems Engineering in the next decades [9]. Therefore, several companies in aerospace and other domains have already started the transitioning to MBSE for the development of *complex systems*.

B. The ambition of model-based system architecting in the AGILE 4.0 project

The transition from a document to a model-based Systems Engineering approach is one of the main ambitions of the EU funded H2020 AGILE 4.0 project (2019-2022) [10], led by DLR, which enlarges the scope of the predecessor EU-H2020 AGILE project by introducing all the development activities of a typical Systems Engineering process and including all the main pillars of the aeronautical supply-chain: design, production, certification and manufacturing. In AGILE 4.0 project, MBSE technologies are leveraged and integrated within an *MBSE development system* for the modeling, assessment, and optimization of *complex systems* addressing the entire life cycle. Therefore, the *MBSE development system* being developed in AGILE 4.0 extends the scope of the *MDO system* built in AGILE project by including all the upstream activities that anticipate the setup and deployment of Multidisciplinary Design and Optimization (MDO) processes, such as definition of *complex systems* (in terms of stakeholders, needs and requirements) and *system architecting* [11], [12]. The *MBSE development system* supports an *architectural framework* to represent through a MBSE approach all the *system architectures*, where "*system*" refers to the *complex system* (i.e. the aircraft).

The *architectural framework* and *MBSE development system* developed within the frame of AGILE 4.0 for the definition and modeling of system stakeholders, needs and requirements has been presented in [13]. The present paper

expands the context of the AGILE 4.0 architectural framework and its implementation by now focusing on the *system architecting* step of a Systems Engineering process. This step aims at generating a large number of potential system architecture that should meet the stakeholder needs and be compliant with the system technical requirements. Any Systems Engineering approach recommends to develop systems along an increasing evolution of the system's level of abstraction. In other words, the stakeholder expectations should entail solution-neutral functions that the system will have to provide through its components, but multiple alternative components can be selected to fulfil the system functions, therefore originating multiple alternatives of system architecture. This "Systems Engineering way of thinking", which entails a direction of the development process from solution-neutral to solution-specific level of abstraction, is a key enabler to originate a large set of conventional but also innovative concepts, which are necessary to face the challenges affecting aviation sector as mentioned earlier. The aim of this paper is therefore to extend the previously presented AGILE 4.0 *architectural framework* in order to guide the designer in the system architecting step of a Systems Engineering Product Development process.

In order to reach the previously stated objectives, this paper is organized as follows. After the introduction of Section I, the *MBSE architectural framework* proposed in this paper is introduced in Section II, where the definitions of the main elements composing an architectural framework are given, namely *process*, *ontology* and *viewpoints*. The detailed description of the *architectural framework* is given in Sections III, IV, and V. Few examples of application of the guidelines provided by the MBSE architectural framework are described in Section VI. The paper ends with Section VII by deriving conclusions and suggesting future developments.

II. A model-based architectural framework for the development of system architectures

As stated in the introduction, this paper focuses on the system architecting activities of a *Systems Engineering Product Development process*. Different sources in literature propose these kinds of process for the development of *complex systems*, part of them in the form of standard (e.g. [14], [15], [16] and [17]), others described in Systems Engineering handbooks (e.g. [18] and [19]). The interested reader can find an overview of these main *Systems Engineering Product Development processes* in [20] and [21].

A novel systems development process combining a typical Systems Engineering approach together with a typical MDO approach is being developed in the AGILE 4.0 project, and schematized in Figure 1 [11].

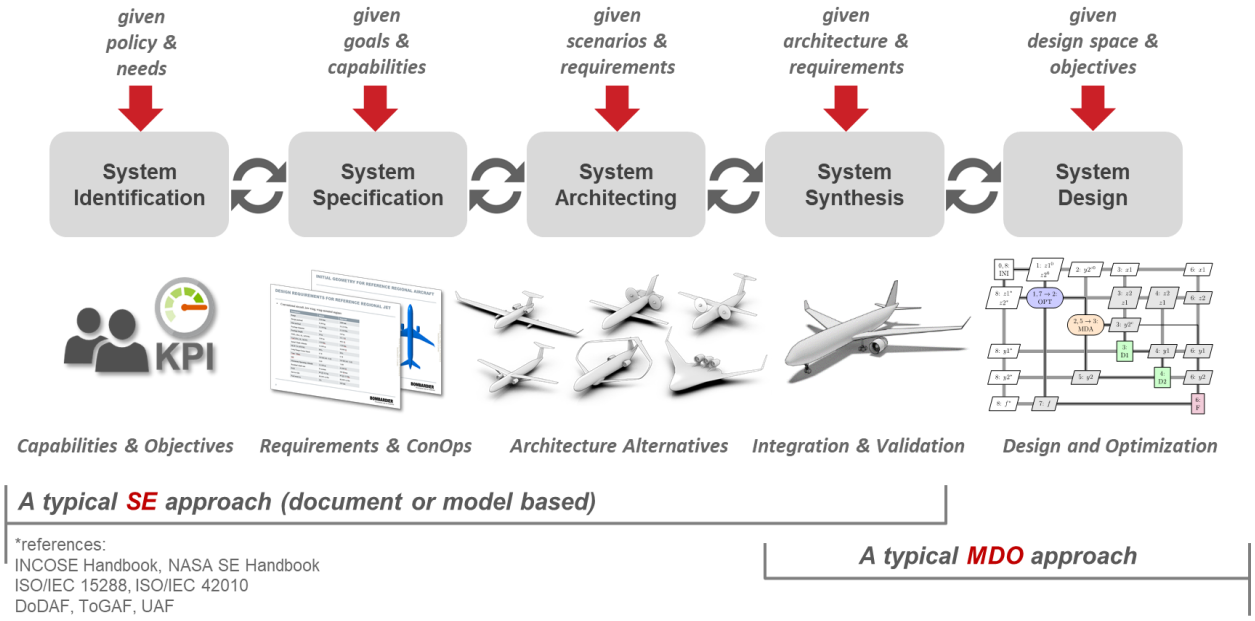

Figure 1 Systems Engineering Product Development process set up in AGILE 4.0 for the collaborative development of complex aeronautical systems [11].

The process of Figure 1 aims at designing and optimizing aeronautical systems (i.e. *System Synthesis* and *System Design* steps), starting from upstream system development activities. The first step aims at defining the systems' required capabilities and objectives. System stakeholders are identified, and their expected needs about the system are collected. Then, stakeholder needs are evaluated through operational scenarios and transformed into system

requirements during the *System Specification* step. The guidelines supporting the designers during the definition and modeling of the main elements (e.g. stakeholders, needs and requirements) of the first two steps of the AGILE 4.0 development approach are provided in [13]. The present paper instead covers in more details the development activities dealing with the *System Architecting* step. These activities aim at defining all the possible components that should be part of the system architectures in order to provide all the system functions as demanded by the functional requirements. Multiple system architectures can be identified in this step, representing conventional but also innovative solutions. Therefore, an Architectural Design Space is generated, from which different system architectures are derived and moved forward along the AGILE 4.0 *Systems Engineering Product Development process*.

As introduced in Section I, an *architectural framework* can guide the designer during a system development process. The *architectural framework* addressed in the present paper specifically focuses on the definition of system architectures. An *architectural framework* is composed by ***ontologies*** and ***viewpoints***, as explained in [22]. **An *ontology* defines all the key concepts composing the *system architecture* and their relationship**. **A *viewpoint* lists all the conventions for the construction, interpretation and use of *system architectures* from the perspective of specific system concerns, named *perspectives***. ***Viewpoints* are prescribed to create *views* for the representation of the *architectures* of the system under development from the different *perspectives***. In addition, the here proposed architectural framework details the system architecting ***process***, which is set up to derive two kinds of system architectures:

- **Functional system architectures**, made of all the functions that the system should fulfil;
- **Logical system architectures**, made of logical components fulfilling all the functions defined in the functional architectures;
- **Physical system architectures**: made of physical components, i.e. instances of the logical components defined in the physical architectures.

The following Section will describe the *process*, *ontology* and the *viewpoints* of the proposed *architectural framework* for the development and representation of architectures of *complex systems*.

III. MBSE architectural framework: *process* for system architecting

This Section presents the process developed in AGILE 4.0 for the architecting of complex systems. As explained before, three kinds of architectures are identified: functional, logical and physical. Therefore, the system architecting process defined in the project can be divided in three parts: functional, logical and physical system architecting.

A. *Functional* system architecting process

The aim of the functional architecting part of the system architecting process (see Figure 2) is to identify all the functions that the system (through its components) should fulfill. 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. Given a System of Interest (or system under design), i.e. a product being developed by a design team, a level of elaboration of it is a specific level in a hierarchy ranging from a top system-level (e.g. aircraft-level) to a lower subsystem-level (e.g. wing-level). Therefore, a boundary function specifies "what" a level of elaboration of the system should provide through its architecture, determined from functional requirements. For example, *provide propulsive thrust* is one of the boundary functions that an aircraft should fulfil through one of its components. Boundary functions are solution-neutral, and they are not determined on the basis of which components are part of the system architecture. Boundary functions of the system level of elaboration can be determined through use cases. A use case is a 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). For example, a use case can address the aircraft high-level function *to fly* in the *cruise* condition. 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) [23], FAS (Functional Architectures for Systems) [24] and SYSMOD [25]. 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.

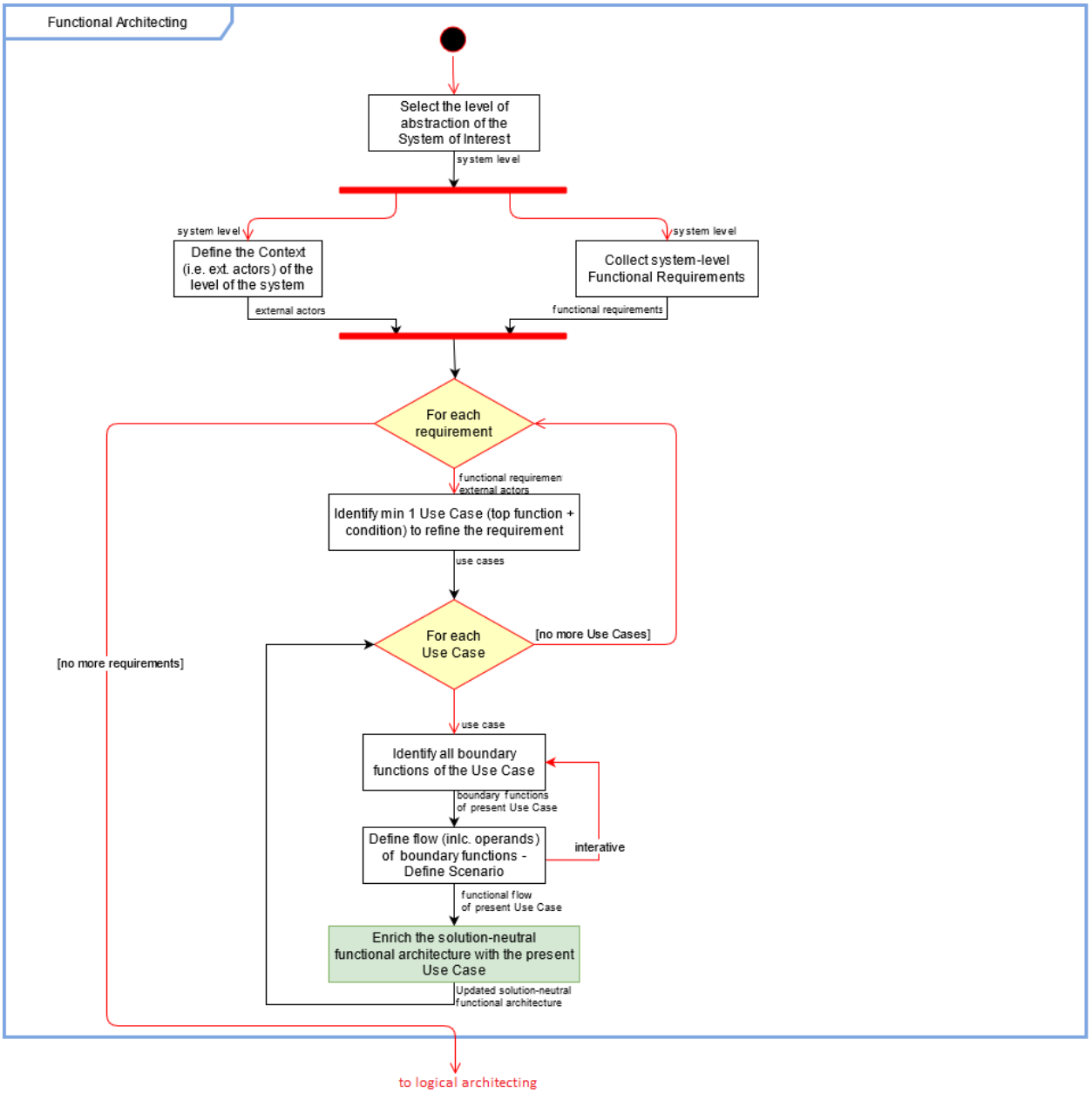

Figure 2 Process of the proposed architectural framework for *functional* system architecting.

B. *Logical* system architecting process

The aim of the logical architecting part of the system architecting process (see Figure 3) is to generate multiple architectures characterized by different logical components fulfilling all the boundary functions defined in the previous part. For instance, the components *turbofan engine* or *turboprop engine* can fulfil the function *provide propulsive power*. Components can also induce other functions – which are therefore 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, quantities of interest 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. More details about the theory behind the logical architecting process can be found in [26].

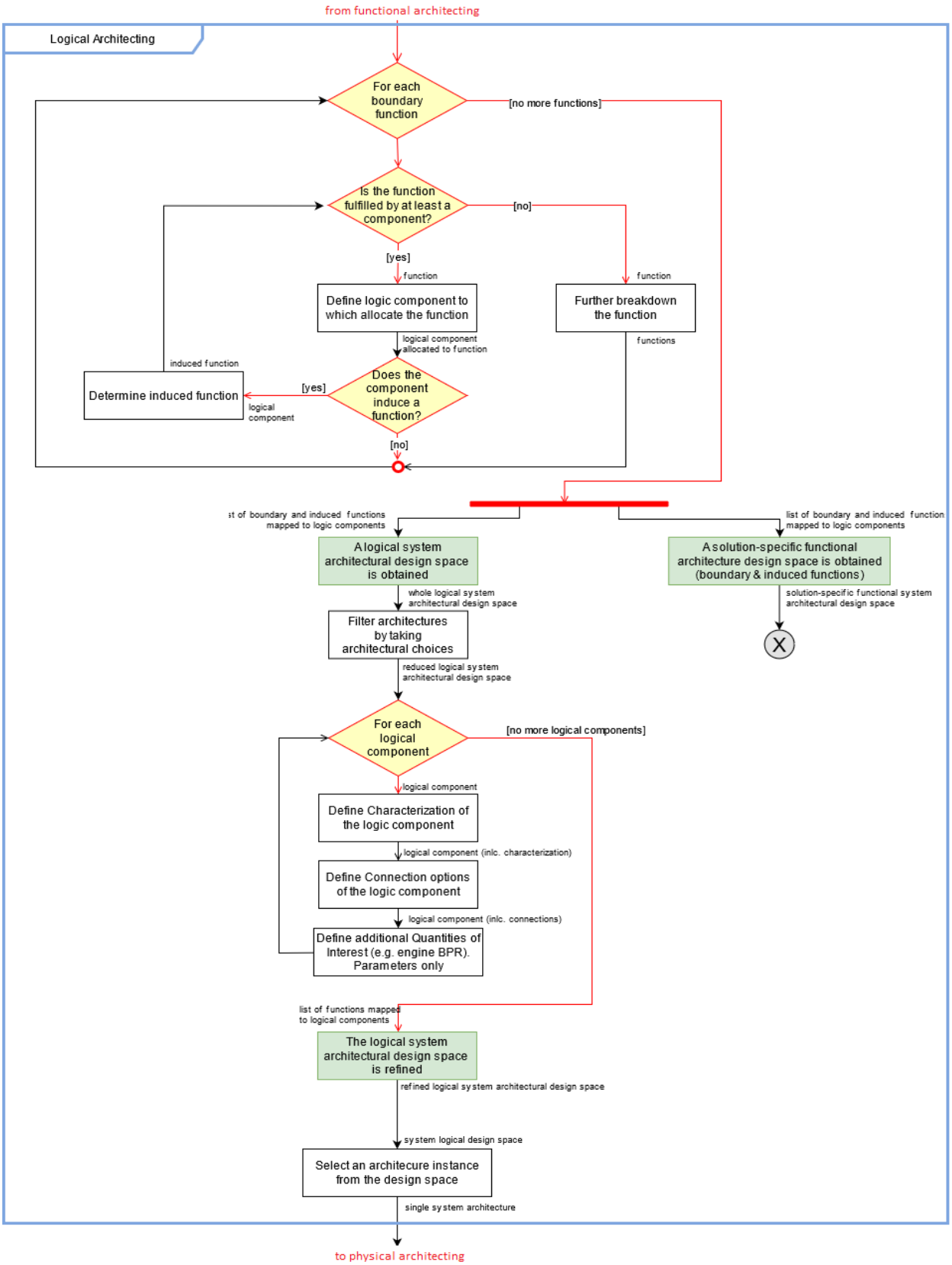

Figure 3 Process of the proposed architectural framework for *logical* system architecting.

C. *Physical* system architecting process

The aim of the physical architecting part of the system architecting process (see Figure 4) is to identify suitable physical components compliant with the non-functional requirements of the system's level of elaboration. A physical component instantiates a logic one by specifying characteristics as performance, dimension, 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 MDO processes. An MDO 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 [27]. Moreover, the mapping between architecture components and disciplinary competences can be done at this stage as recommended in [28], 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. Once an MDO 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.

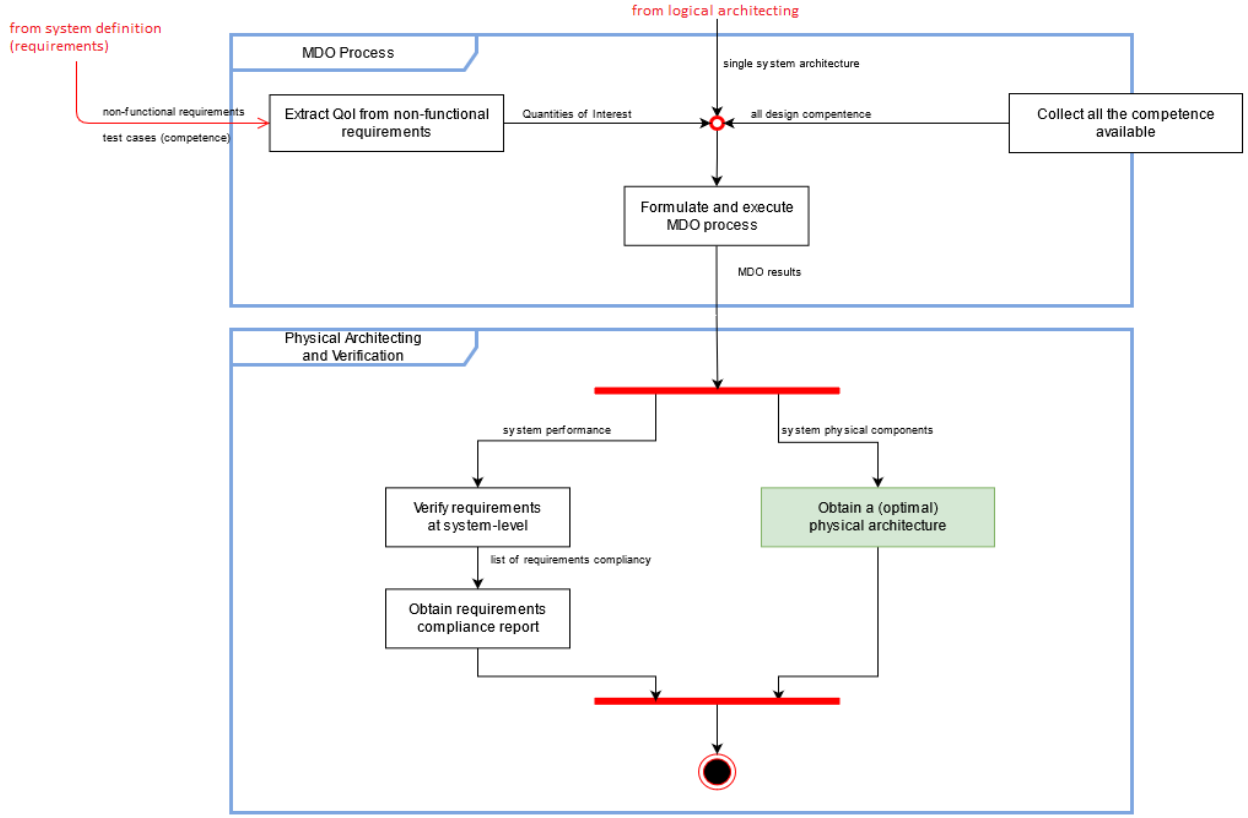

Figure 4 Process of the proposed architectural framework for *physical* system architecting.

IV. MBSE architectural framework: *ontology* for system architectures

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 Figure 5. The ontology is represented in SysML (System Modeling Language) [29], specifically through a SysML Internal Block Diagram, where the new stereotype named «*ontology concept*» is derived from the SysML element «*Block*» and it represents each main architecture element.

A 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 **functionalities**, each one of them expressed by a **functional requirement**. These types of requirement are refined by **use cases**, which describe how **external actors** are involved with the level of abstraction of the system. In addition, use cases are made of multiple **boundary functions**, which is a specialization of a **function**. 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**.

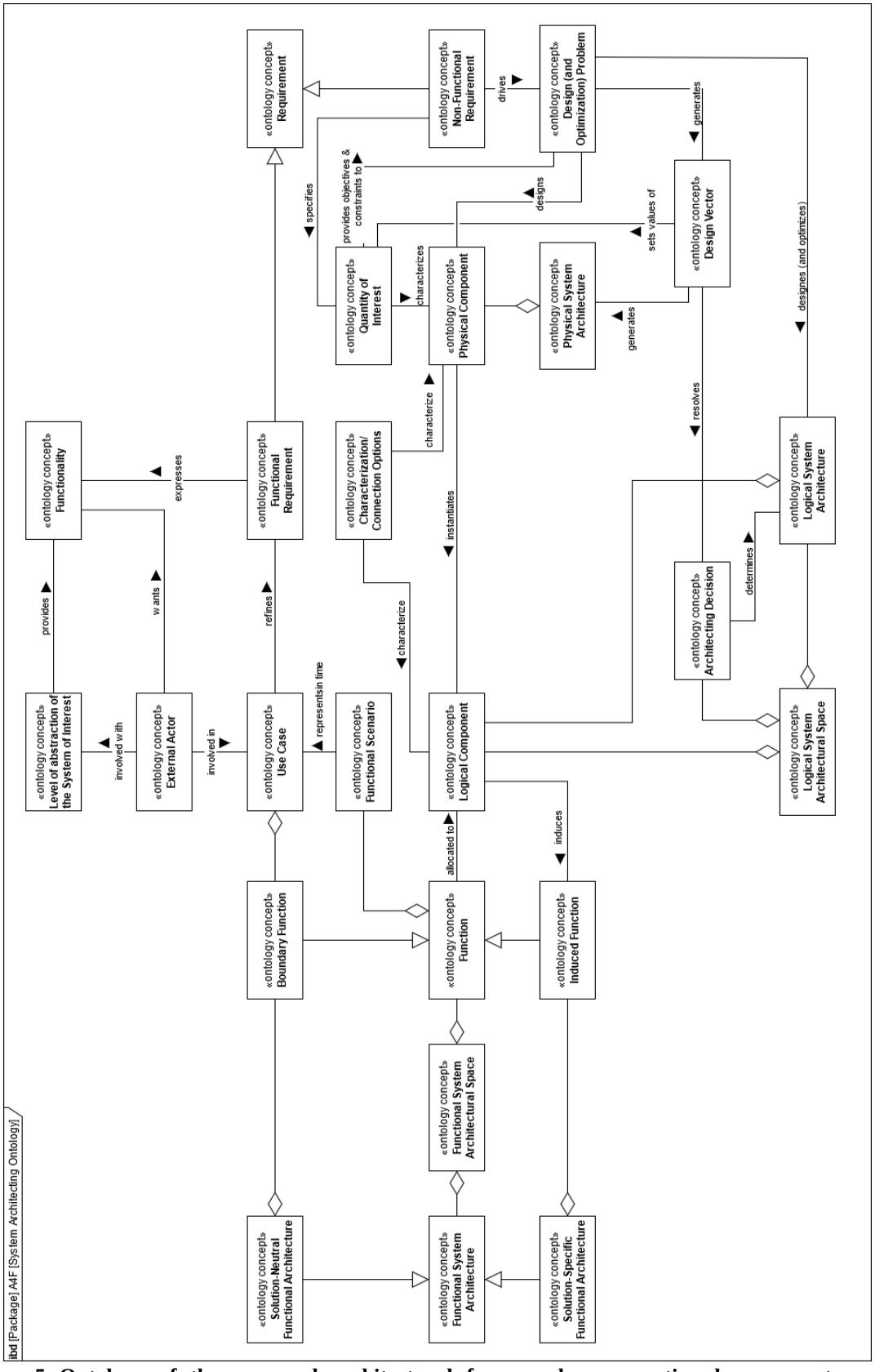

Figure 5 Ontology of the proposed architectural framework representing key concepts and their relationships about system architectures.

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 include also 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**.

Each logical architecture is then **designed and optimized** through an MDO problem, which is driven by **quantities of interest** (e.g. constraints) extracted from the **non-functional requirements** and included in **design vectors**. The results that are derived from the execution of MDO 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**.

V. MBSE architectural framework: *viewpoints* for system architectures

Multiple viewpoints are part of the MBSE architectural framework for the representation of the system architectures model. 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 [30] (implemented in the tool Capella) is used as well in some viewpoints. Finally, the Architecture Design Space Graph (ADSG) [26] is employed as language to model architectural design spaces.

A. Functional Architectures *viewpoints*

During the system functional architecting process, four view can be created by adhering to the viewpoints described in the present subsection and summarized in the SysML Package diagram of Figure 6.

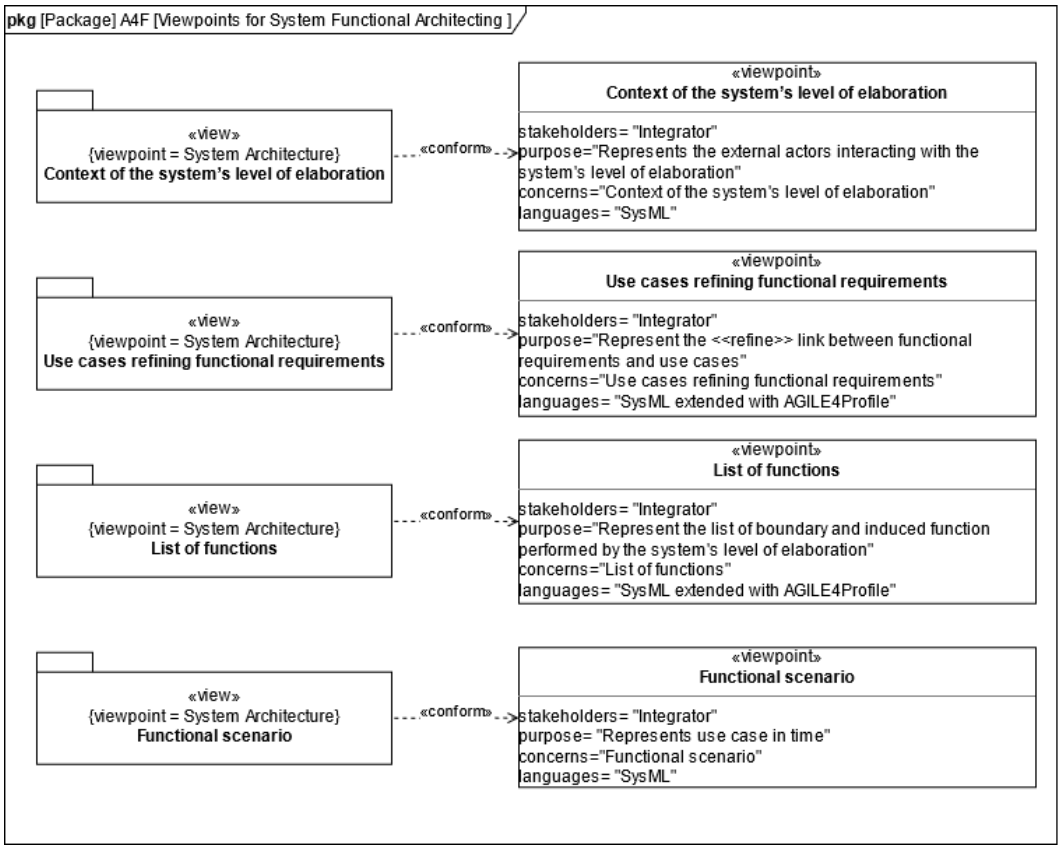

Figure 6 Proposed viewpoints for the representation of system functional architectures.

1. *Viewpoints "Context of the system's level of elaboration" and "Use cases refining functional requirements"*

The first two viewpoints proposed in this paper make use of the SysML diagrams of Figure 7, and they represent the interactions of the system's level of elaboration with external actors and the refinement of functional requirements by means of use cases.

The viewpoint "*Context of the system's level of elaboration*" can be represented by the SysML Use Case Diagram of Figure 7 (a). This representation is commonly used in MBSE activities (e.g. [25]) in order to show which external actors (e.g. other systems, stakeholders) interact with the system's level of elaboration in each use case. As an original contribution, the new stereotype «*systemStakeholder*» defined in the AGILE4Profile (see the Appendix for more details) can be included in the view. This new element represents every stakeholder (e.g. pilot, passenger, maintainer) that interacts with the system of interest when in operation.

A SysML Requirement Diagram like the one of Figure 7 (b) can be employed to represent which use cases refine which functional requirements. Again, a new stereotype name «*requirementPlus*» is defined in the AGILE4Profile, and it is used to model each system requirement (more information can be found in [13]).

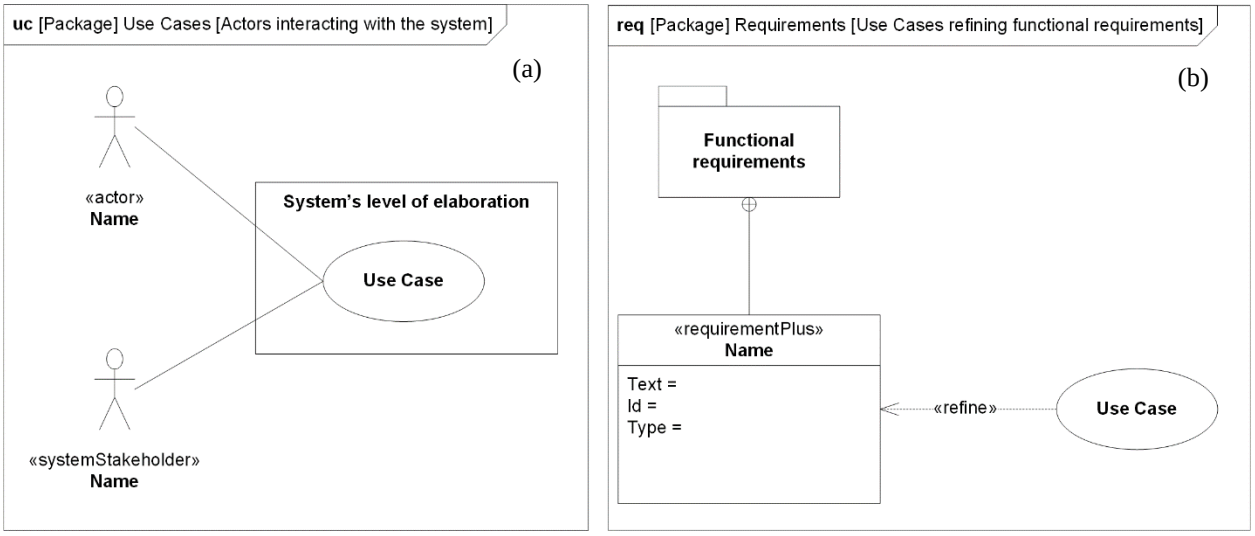

Figure 7 (a) SysML Use Case Diagram representing the context of the systems level of elaboration; (b) SysML Requirement Diagram representing Use Cases that refine functional requirements.

2. *Viewpoint "List of functions"*

A SysML Block Definition Diagram (see Figure 8) can be made to display the functions that are included in one or more use cases, according to the viewpoint "List of function". This view utilizes the new stereotype «*function*» defined in the AGILE4Profile to represent both boundary and induced functions. Functions can be represented in a hierarchy through the "composite association" relationship (i.e. connector with the black diamond). In this way, it is possible to indicate which lower-level functions are derived from a top function.

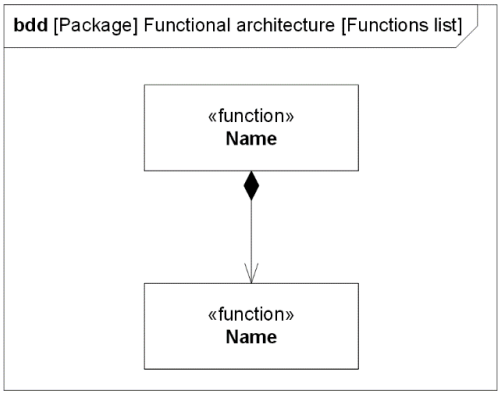

Figure 8 SysML Block Definition Diagram representing the list of system functions.

3. *Viewpoint "Functional scenario"*

The determination of the boundary functions that are included in a use case can be effectively done by identifying functional scenarios. The approach here proposed entails the definition of a single functional scenario per use case. A different recommendation is instead proposed by Eriksson *et al.* [23], who suggest to evaluate each use case in up to three different types of scenario, each one representing a condition: "main success", "alternative" and "exceptional" scenarios. However, the approach proposed in this paper evaluates each top function – therefore not a use case – in different conditions (nominal and off-nominal), therefore deriving multiple use cases described by single functional scenarios. Views representing functional scenarios can be built through two kinds of SysML diagram: Activity and Sequence, as depicted in Figure 9 (a) and (b) respectively. The SysML Activity Diagram of Figure 9 (a) models a single high-level function, and it represents the lower-level functions that happen in different conditions. The SysML element "Decision Node" (i.e. a white rhombus) is used to differentiate the conditions. The top function evaluated in the different conditions entails multiple Use Cases, which are represented in time through the SysML Sequence Diagram of Figure 9 (b). It should be noted that other alternatives are present in literature for the modeling of the functions belonging to a Use Case. Traditionally, a SysML Activity Diagram is utilized for this purpose (e.g. see [24]). Eriksson *et al.* [23] instead propose to describe Use Case through natural language, although they mention SysML Sequence Diagram as a graphical way for representing the sequence of functions. However, it is here recommended to use a diagram rather than natural language for the description in time of Use Case, therefore fostering the transition towards MBSE.

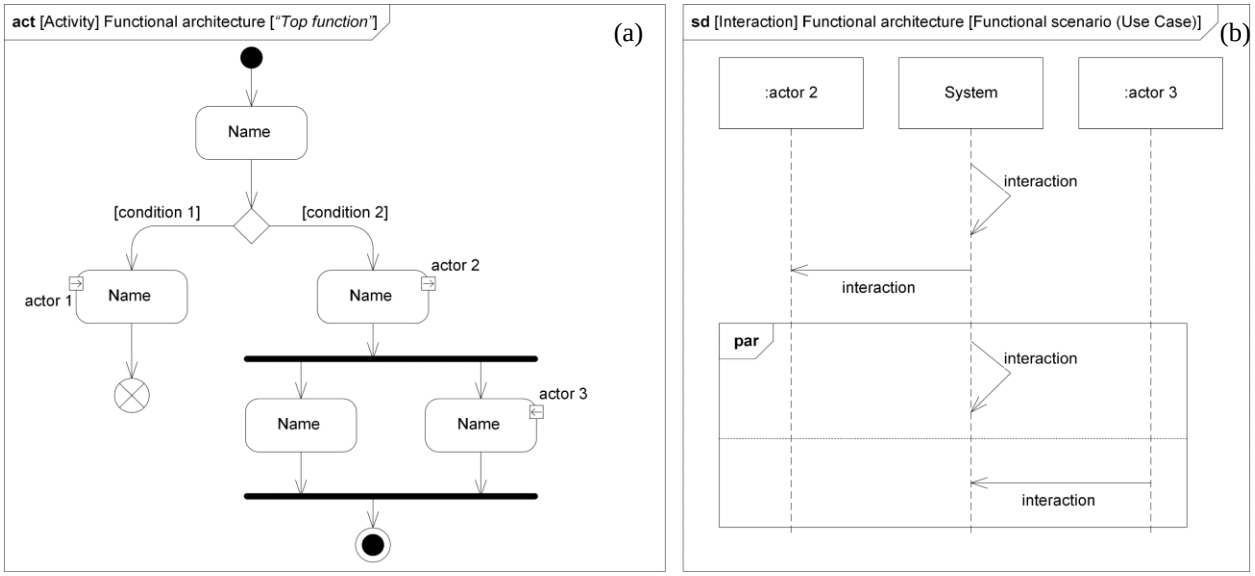

Figure 9 (a) SysML Activity Diagram representing the functions derived from a top function and their relations in multiple conditions; (b) SysML Sequence Diagram representing a single Use Case in time (i.e. functional scenario).

B. Logical Architectures *viewpoints*

The present paper proposes five viewpoints to represent logical architectures, as described below.

1. *Viewpoint "Allocation functions – logical components"*

The "*allocation functions – logical components*" is one of the views that can be created in the Logical Architecting. Two solutions are here proposed, each one exploiting a different modeling language. The former makes use of the Domain Specific Language of the ARCADIA method, and it results in the *[LAB] Logical Architecture Diagram*, a template of which is depicted in Figure 10. The blue rectangles represent system logical components (dark blue) and external actors (light blue), while the (boundary and induced) functions are represented by the green rectangles. Green rectangles are included into the blue ones, in order to depict the allocation of functions to logical components. Furthermore, this view shows the functional interactions (named *Functional Exchanges*) between system functions.

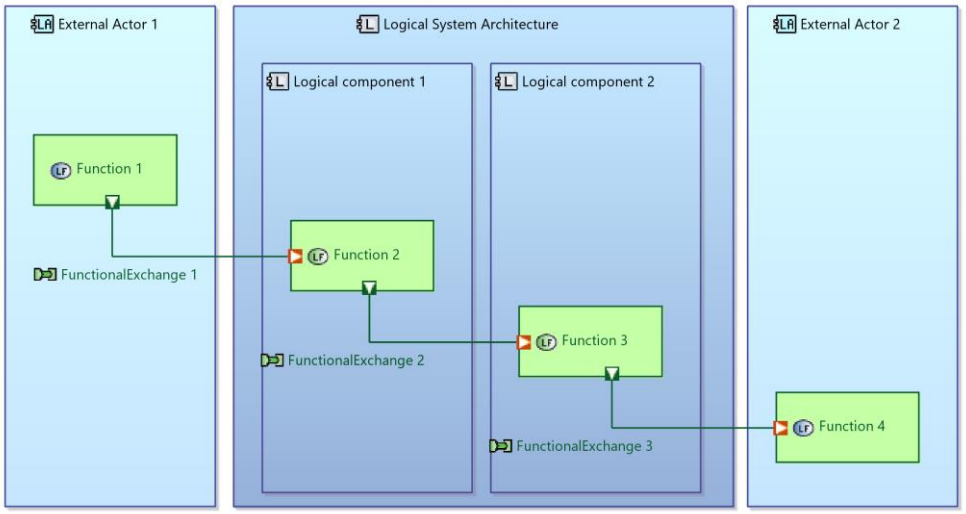

Figure 10 [LAB] Logical Architecture Diagram of the ARCADIA method representing the allocation functions – logical components.

The second solution is a *SysML Activity Diagram* (see Figure 11), where functions are represented by rectangles, which are allocated to components through *swimlanes*, vertical divisions of the diagram, each one per logical component. This view can be derived from the SysML Activity Diagram built to represent functional scenarios (see Figure 9 a).

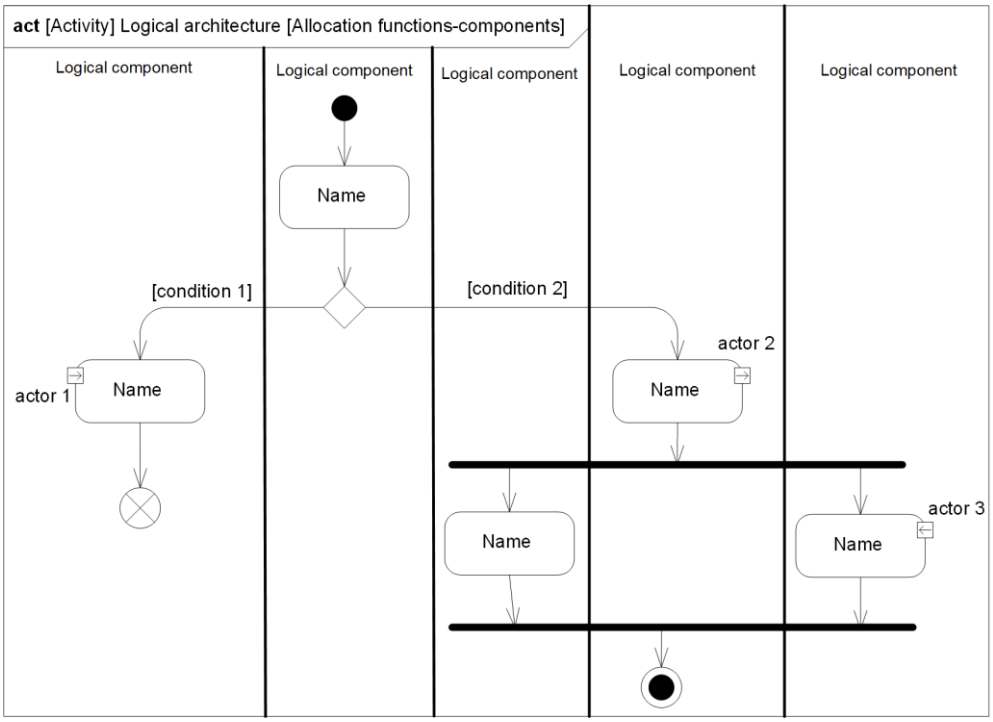

Figure 11 SysML Activity Diagram representing the allocation functions – logical components.

2. Viewpoint "Architecture Design Space"

The ADSG is proposed as modeling language in the MBSE architectural framework to model the Architecture Design Space. A template of view is depicted in Figure 12, which shows all the logical components that can be part of the system architecture in order to provide all the system functions, both induced and boundary ones.

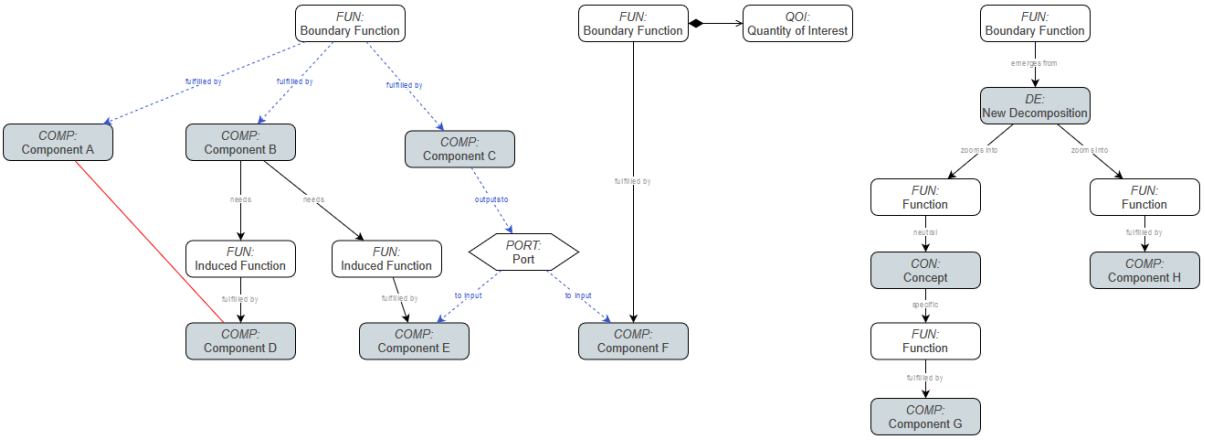

Figure 12 Graph representing the template prescribed by the *viewpoint* "Architecture Design Space" to show logical components and functions and their relations (i.e. "fulfilled by" and "needs").

3. *Viewpoints "Logical components of an architecture" and "Connections between logical components in an architecture"*

Multiple architecture instances can be generated from the Architecture Design Space by taking architectural decision, e.g. by selecting a specific component that provides a function. Two viewpoints are proposed to represent information for each specific logical architecture.

The viewpoint "*Logical components of an architecture*" can be used to represent through a SysML Block Definition Diagram like the one of Figure 13 (a) which logical components are part of the system's level of elaboration. As original contribution, the new stereotype «*logicComponent*» defined in the AGILE4Profile is included in any view built according to this viewpoint. This new SysML element displays the functions the components fulfil, and the characterization and connection options of each component.

The viewpoint "*Connections between logical components in an architecture*" instead consists of a SysML Internal Block Diagram (see Figure 13 (b)), representing the connections between logical components of the system's level of elaboration and the external actors (both as inputs and outputs). These connections marked in the figure as *Item flow* can be different kinds of exchange, e.g. mechanical power, electric signal, force exchanges.

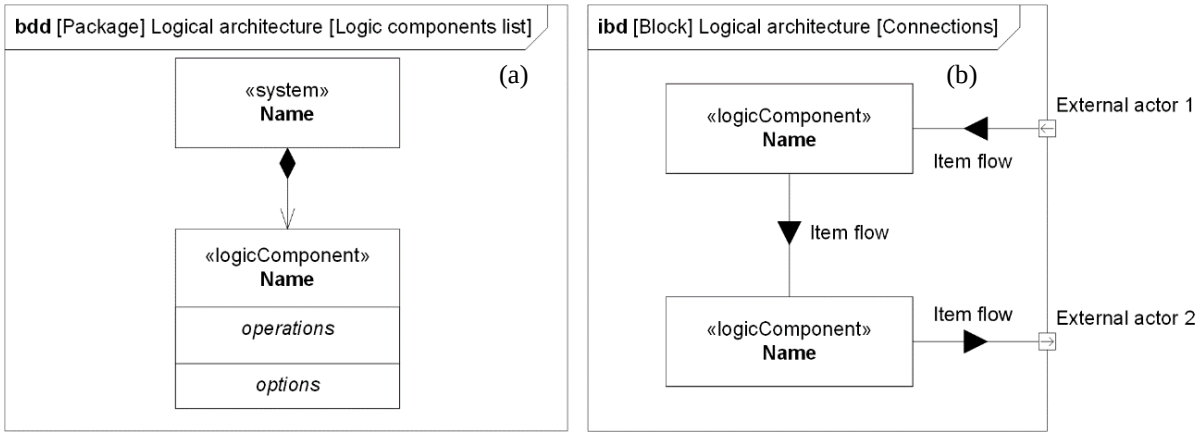

Figure 13 (a) SysML Block Definition Diagram representing the logical components of a specific logical architecture; (b) SysML Internal Block Diagram representing the connections between logical components of a specific logical architecture.

4. *Viewpoint "States of the logical components"*

The last viewpoint of the Logical Architecting part of the system architecting process can be built to model the states of each logical component, and the transitions from one state to another. A state represents some significant condition in the life of a block (i.*e. a system component*), typically because it represents some change in how the block

responds to events and what behaviors it performs [8]. The SysML State machine Diagram of Figure 13 is a template that can be used to represent this view.

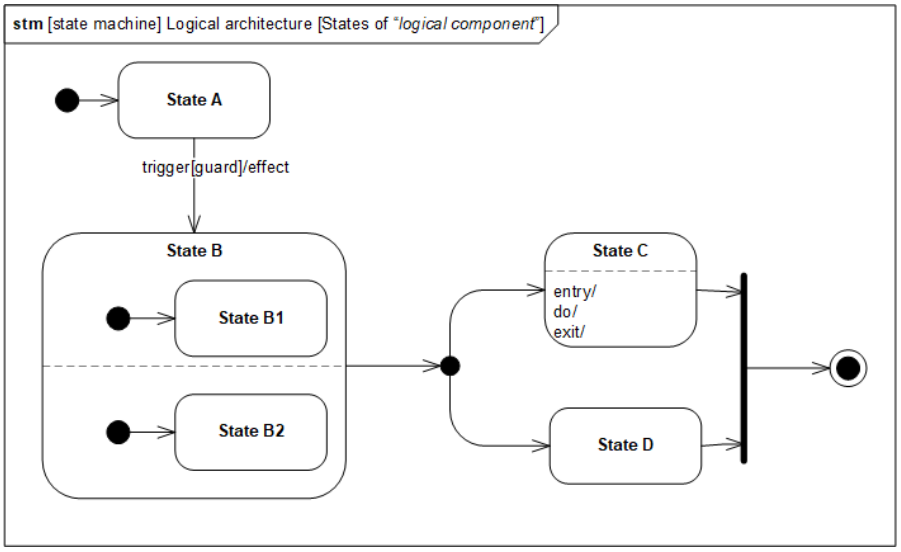

Figure 14 SysML State Machine Diagram representing the states of logical components.

C. Physical Architectures *viewpoints*

Finally, five viewpoints are recommended by the authors to represent physical architectures. These viewpoints are described below.

1. *Viewpoints "List of physical components" and "Physical Components satisfying non-functional requirements"*

The first viewpoint addressed in the present paper about Physical Architecting shows the list of physical components of a physical architecture. The SysML Block Definition Diagram of Figure 15 (a) represents a template that can be employed to represent this view. This diagram includes the new SysML element *physicalComponent*, introduced with the AGILE4Profile as an extension of the *block* element, where the new property *logical component* has been added in order to state which logical component (e.g. an engine) is instantiated by which physical component (e.g. CFM56-5B). The *values* properties represent instead the Quantities of Interest of the system component.

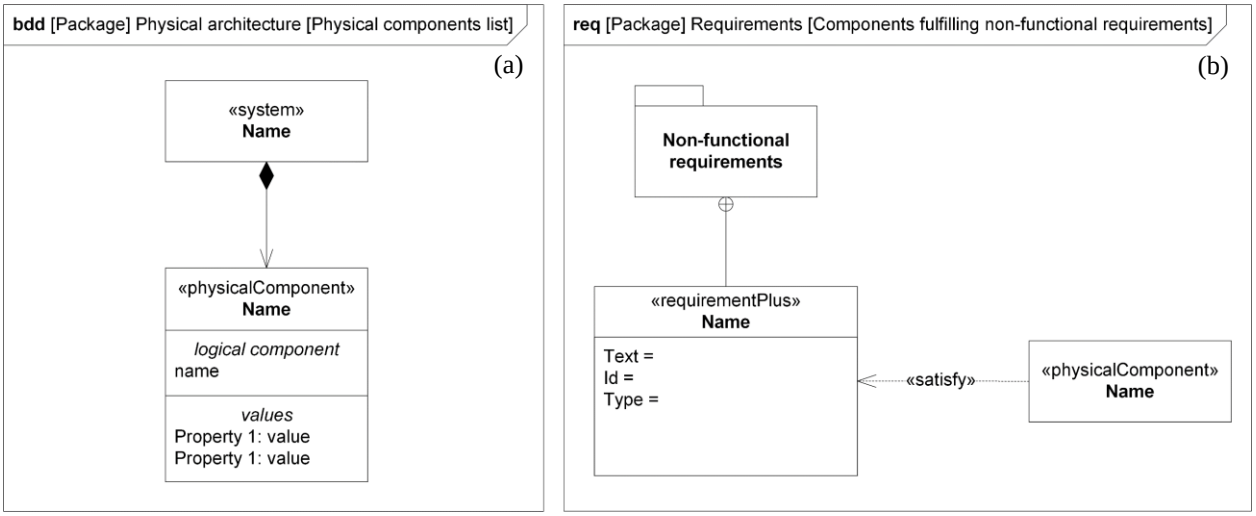

Figure 15 (a) SysML Block Definition Diagram representing the physical components of a specific physical architecture; (b) SysML Requirement Diagram representing the "satisfy" relationships between physical components and requirements.

Physical components can also be linked to requirements, in case they satisfy what stated by requirement statements. The viewpoint "*Physical Components satisfying non-functional requirements*" aims at representing this information. A traditional approach through SysML is characterized by the link between a *block* and a *requirement* SyML element through the satisfy relationship. In this paper instead, the *block* element is substituted with the «*physicalComponent*» stereotype, as illustrated in the SysML Requirement Diagram of Figure 15 (b).

2. *Viewpoint "Connections between physical components in an architecture"*

The Domain Specific Language of the ARCADIA method is instead exploited for the following viewpoint, which aims at representing the connections between the physical components of a physical architecture. The *[PAB] Physical Architecture Diagram* of Figure 16 can be used as template for creating views that conform to this viewpoint. The diagram contains all the elements (e.g. logical components and functions) of the Logical Architecture viewpoint "*allocation functions – logical components"* (see Figure 10). In addition, the [PAB] diagram depicts physical components (yellow rectangles), while connections between them are represented by the *Physical Link* elements.

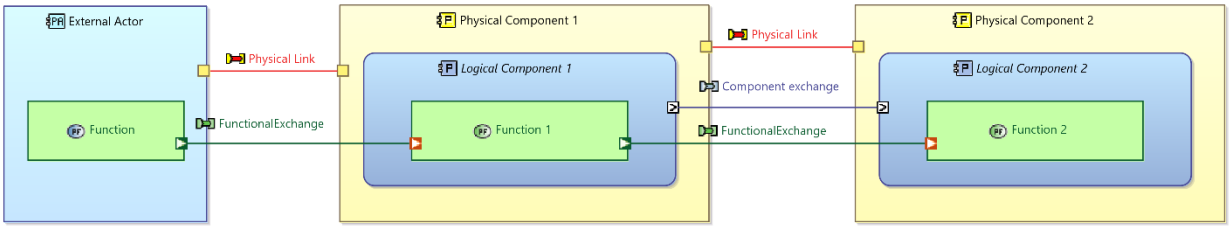

Figure 16 *[PAB] Physical Architecture Diagram* of the ARCADIA method connections between physical components.

3. *Viewpoint "States of physical components"*

The states of physical components and the transitions from one state to another can be represented by a SysML State Machine Diagram, similar to the one for the viewpoint "*States of the logical components*" adressing logical components (see Figure 14). In the case of physical components, however, this kind of diagram can include parameters, as shown in Figure 17. These parameters characterize the transition between states, and they include:

- Time values or variables that trigger the change of state;
- Condition that should be verified in order to be possible a change of state;
- Result determined as a consequence with a change of state.

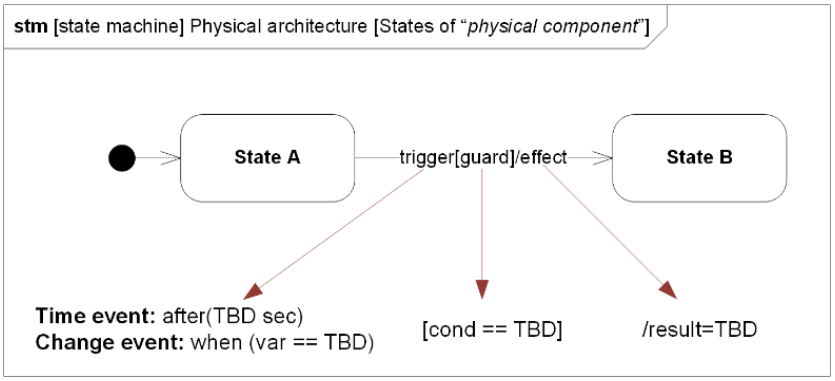

Figure 17 SysML State Machine Diagram representing the states of physical components.

4. *Viewpoint "Interactions between physical components"*

The last viewpoint addressed in this paper aims at representing the interactions between physical components. The SysML Sequence Diagram of Figure 18 can be used for this purpose. This diagram is the same one of Figure 9 (b), but with two main differences. First, each physical component of the system's level of elaboration is depicted in this viewpoint, while in the Functional Architecting case there's only a single block representing the entire system. Therefore, interactions flow between system and actors but also between components. Second, interactions between

physical components are specified by variables. For example, an interaction might be an exchange of power between two components, and the amount of power exchanged can also be indicated in this viewpoint.

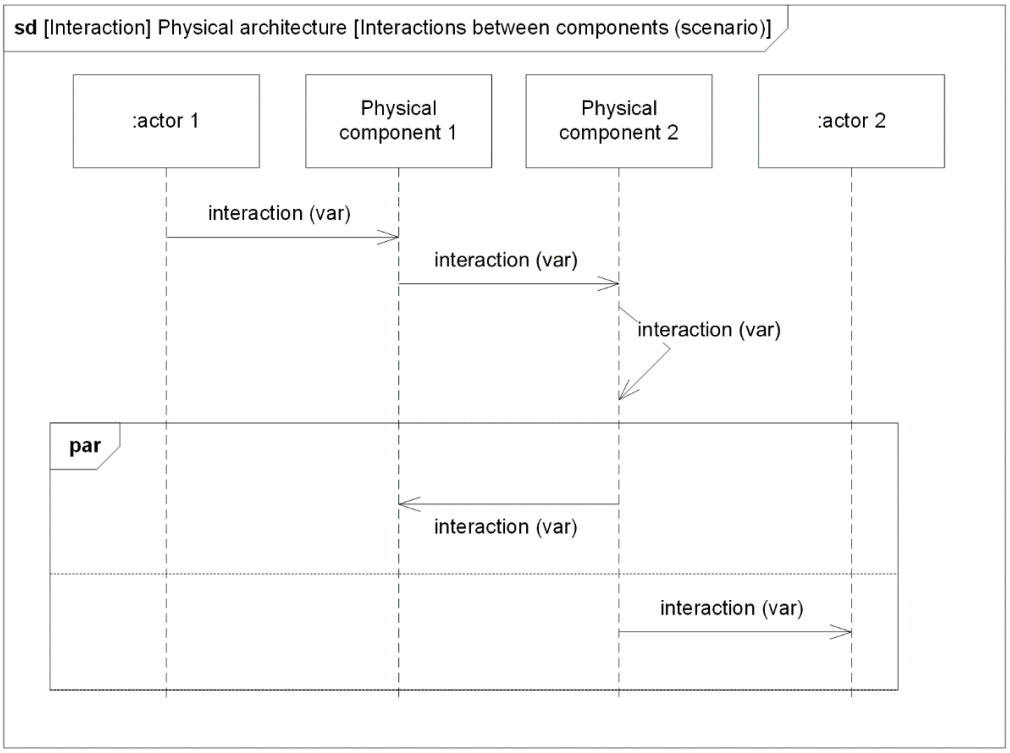

Figure 18 SysML Sequence Diagram representing the interactions between physical components and external actors.

VI. Examples of application of the MBSE architectural framework for system architecting

In the context of the AGILE 4.0 project, the guidelines proposed in this paper for the system architecting have been implemented in the so-called Operational Collaborative Environment (OCE), which integrates several tools developed by the partners of the project Consortium to support all the development activities of the Systems Engineering Product Development process set up in AGILE 4.0 for the collaborative development of complex aeronautical systems. More details about the technologies integrated in the OCE can be found in [31].

The proposed MBSE architectural framework is being exploited through the OCE for the development of different AGILE 4.0 application cases, which are introduced in [12]. Therefore, the proposed MBSE architectural framework is leveraged to support:

1. the design of different architectures of trailing edge flaps including manufacturing constraints [32];
2. the development of different architectures of Horizontal Tail Plane designed and produced by multiple configurations of Supply Chain system [33];
3. the design of conventional and innovative on-board system architectures of a regional aircraft including certification constraints [34];
4. the design of a Unmanned Aerial Vehicle (UAV) accounting for certification constraints [35];
5. the retrofitting of a regional jet [36];
6. the design a family of business jets [37].

Nevertheless, few examples of application of the proposed MBSE architectural framework are presented in this section. More specifically, three examples are proposed, each one of them focusing on a specific part of the system architecting process.

A. Functional Architecting *example of application*

The first example of application focuses on the functional architecting of a generic regional jet aircraft. The functional architecting process represented in Figure 2 is followed in order to determine the boundary functions that the aircraft should fulfil. Once all the functional requirements at aircraft-level are collected, use cases are identified to

refine the top function of each requirement by specifying a condition. The SysML Requirement Diagram of Figure 19 shows some functional requirements, each one of them refined by one or more use cases. For instance, two use cases refine the requirement “The aircraft shall perform manoeuvres”, by specifying the two conditions “in flight” and “on ground”.

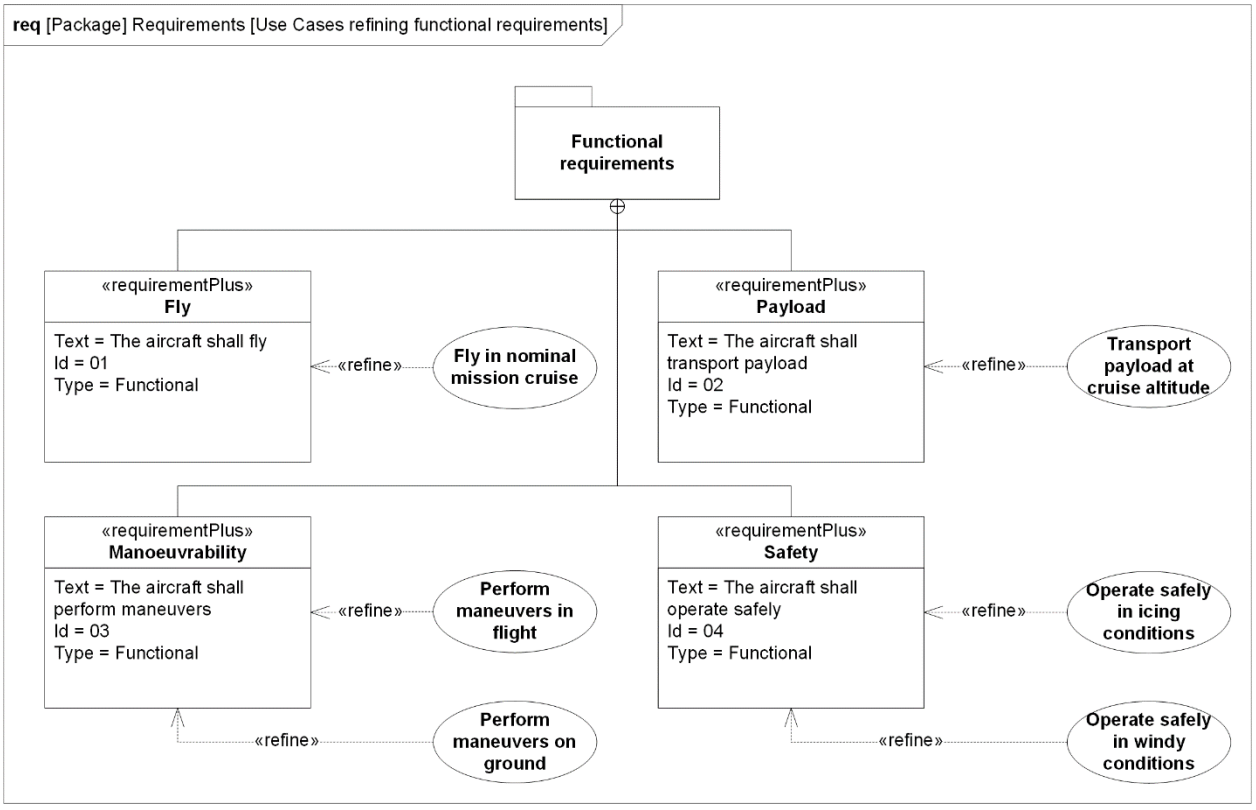

Figure 19 SysML Requirement Diagram representing the use cases that refine some functional requirements of a regional jet aircraft.

The use cases are then utilized to identify all the boundary functions that the regional jet aircraft should fulfil. The identified functions are collected in the SysML Block Definition Diagram of Figure 20. For example, the use case “*fly in nominal mission cruise*” entails the boundary functions “*to generate lift*” and “*to provide thrust*”.

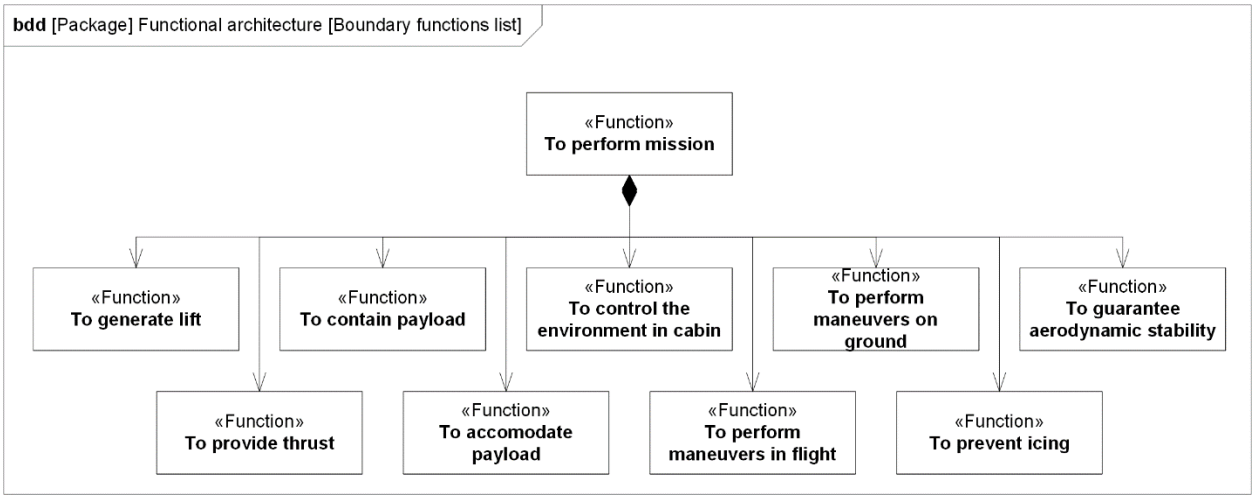

Figure 20 SysML Block Definition Diagram representing some system functions of a regional jet aircraft.

B. Logical Architecting *example of application*

A different system's level of elaboration is here used to present an example of the view "Architectural Design Space", conforming to the relative viewpoint of the logical architecting part. The system to be architected is an engine, and its architectural design space is modeled and represented as ADSG in Figure 21. In this case, the boundary function is "*Accelerate Air*", as this is the only function that is not induced by any of the components. This boundary function can then be fulfilled by two components: the *Fan Compressor* or the *Nozzle*. In both cases, additional functions are induced, which have to fulfill by additional components. For example, in case a Nozzle is selected in an architecture instance, a Burner should be introduced as well in order to fulfill the function "Energize air". In turns, the Burner induces the function "Compress air", which requires a Compressor. Finally, a Turbine has to be introduced to provide the mechanical power required by the Compressor. More information about the present example can be found in [28].

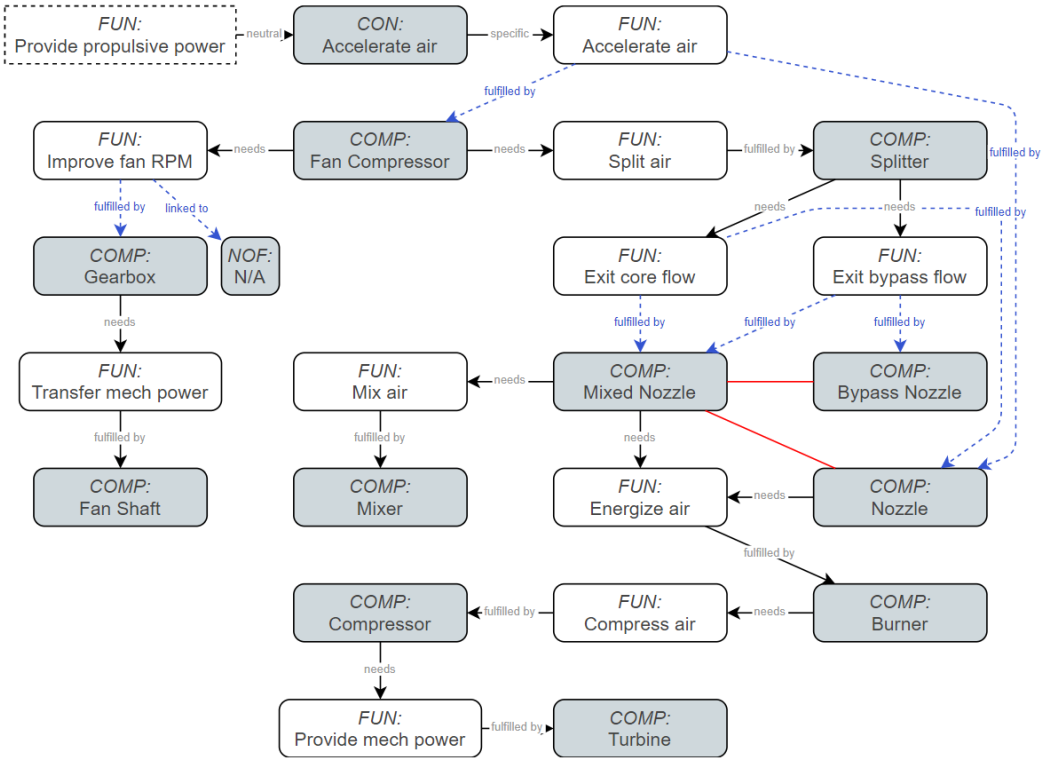

Figure 21 Graph representing the Architecture Design Space of a jet engine [28].

C. Physical Architecting *example of application*

The last view proposed as an example represents the connection between the physical components of a simplified communication system. This physical architecture is made of two physical components: a real headset and a real radio. The Physical Architecture Diagram of the ARCADIA method represented in Figure 22 shows this view.

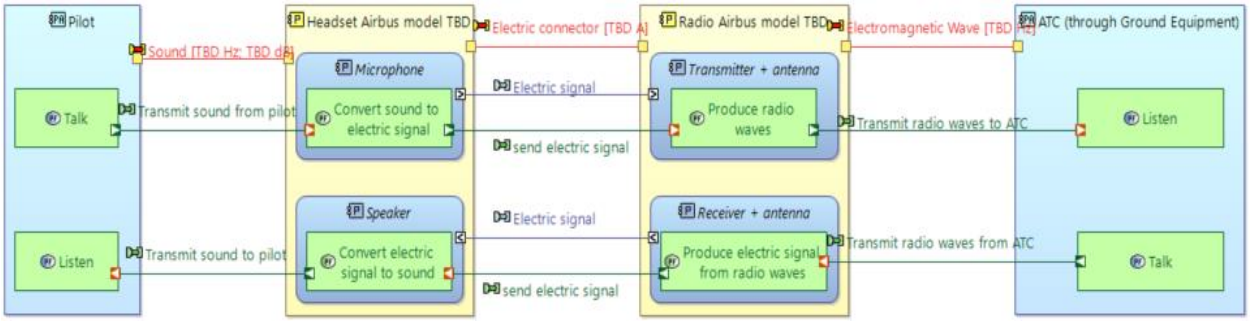

Figure 22 [PAB] Physical Architecture Diagram of the ARCADIA method connections between physical components of a simplified communication system.

The pilot and the Air Traffic Control (ATC) need to communicate (functions *talk* and *listen*). The sound emitted by the pilot – which is characterized by TBD Hz frequency and TBD dB amplitude – is received by a microphone (*logical component*), which is part of the headset. The microphone converts the sound into an electric signal, which is sent to the radio via an electric connector or wire (*physical link*). This electric signal then generates electromagnetic waves that are transmitted to the ATC. The opposite happens in the other direction, i.e. when the ATC talks to the pilot. In this case, the radio waves are transmitted to the receiver of the radio mounted on the aircraft, which converts this wave into an electric signal that is converted in sound and sent to the pilot via the speaker of the headset.

VII. Conclusions and future developments

A new architectural framework that leverages an MBSE approach is being realized within the context of the EU-funded research project H2020 AGILE 4.0. This MBSE architectural framework offers guidelines to designers in the development of complex aeronautical systems. These guidelines aim at accelerating, improving and streamlining all the development activities of a typical Systems Engineering process, as stakeholder needs identification, transformation of needs in technical requirements, generation of multiple alternative system architectures, design and optimization, trade-off assessments, decision-making and validation and verification activities.

More specifically, present paper focuses on the system architecting part of the MBSE architectural framework. Therefore, guidelines supporting the generation of conventional and innovative system architectures starting from the system functional requirements are provided. These guidelines include a process defining all the steps necessary to generate system architectures, an ontology providing the definition and relationships between different concepts related to system architecting, and multiple viewpoints that can guide the representation of the information regarding the system architectures.

Further developments will entail the formalization of the latest activities of AGILE 4.0 Systems Engineering Product Development process, therefore supporting activities related to trade-off assessment and decision making.

Appendix

A new profile derived from SysML and named *AGILE4Profile* has been developed, and it is represented in the *SysML Package Diagram* of Figure 23.

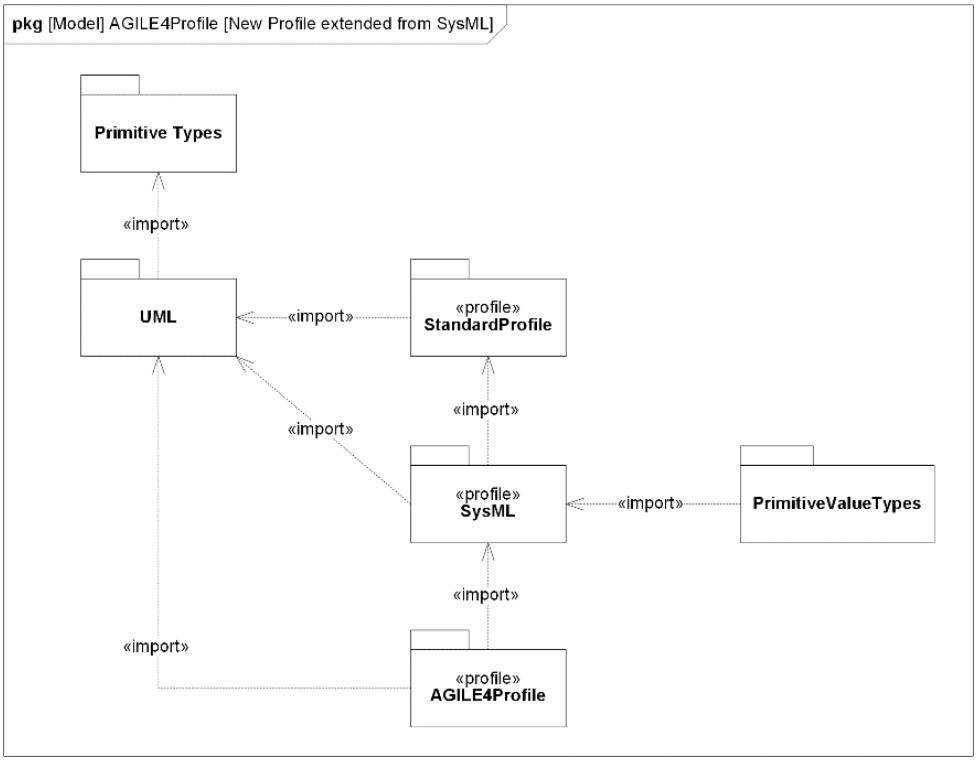

Figure 23 *SysML Package Diagram* representing the extension of the SysML profile (*AGILE4Profile*) adopted in the proposed *architectural framework* for the modeling of system architectures.

The new stereotypes introduced with the *AGILE4Profile* are instead collected in the *SysML* diagram of Figure 24. The new profile has been illustrated in [13]. The new stereotypes exploited in the present paper are marked in orange in the figure and illustrated below.

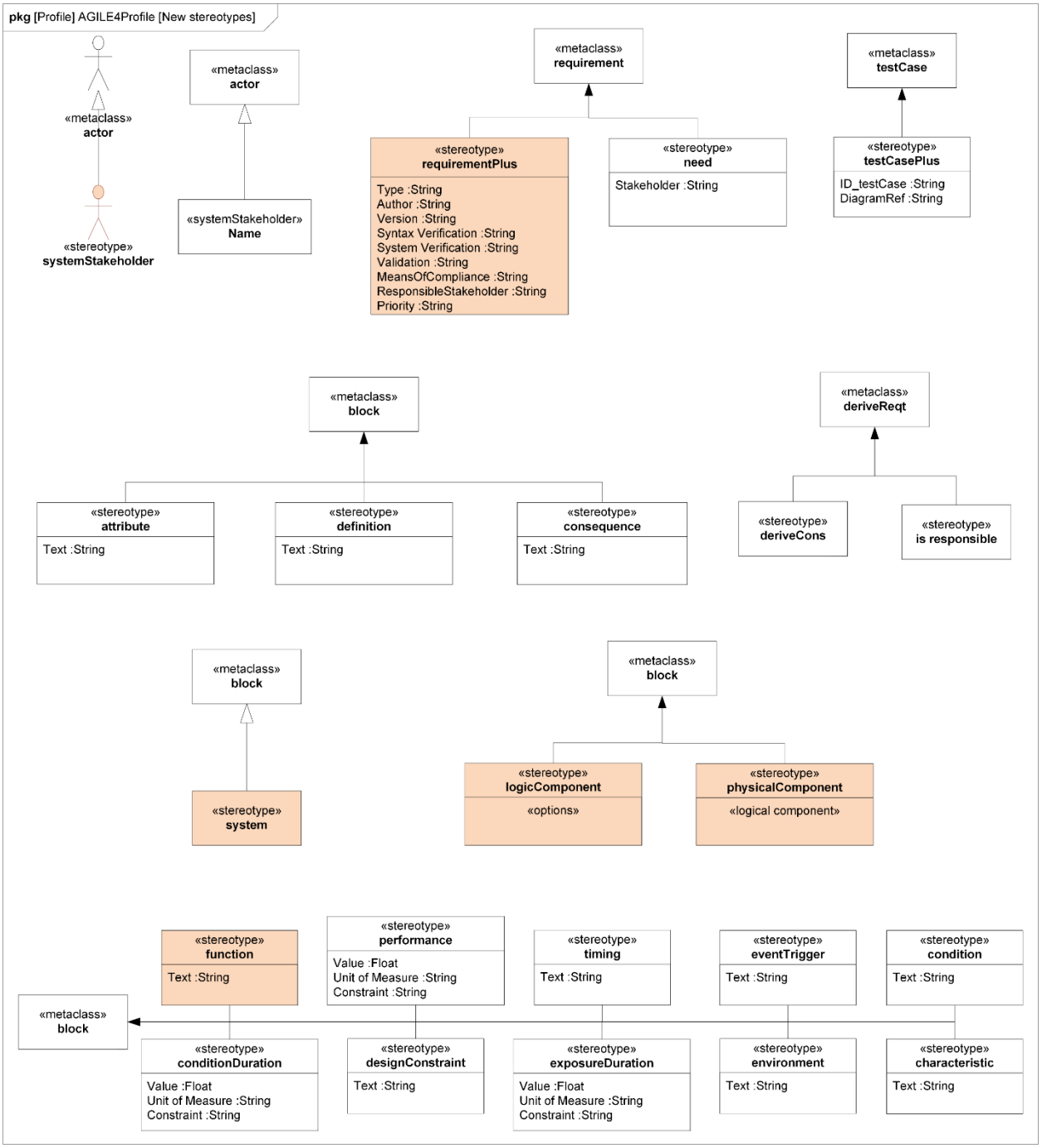

Figure 24 *SysML Package Diagram* representing the new stereotypes for the modeling of stakeholders, needs and requirements introduced with the new *AGILE4Profile*.

The following extensions have been made to the standard SysML and are part of the *AGILE 4Profile*:

- The new stereotype ***«systemStakeholder»*** specializes the metaclass «*actor*».

- The metaclass «*requirement*» is employed to generate the stereotype ***«requirementPlus»***, which other than specifying the text and ID of the requirement as prescribed by SysML, includes additional properties, for instance type of requirement (e.g. functional or performance), author, version and verification status.
- The metaclass «*block*» is specialized with the new stereotypes ***«system»*** and ***«function»***.
- The new stereotypes ***«logicComponent»*** and ***«physicalComponent»*** are derived from the metaclass «*block*».

Acknowledgments

The research presented in this paper has been performed in the framework of the AGILE 4.0 project (Towards cyber-physical collaborative aircraft development) and has received funding from the European Union Horizon 2020 Programme under grant agreement n◦ 815122. The authors are grateful to all the partners of the AGILE 4.0 Consortium for their contribution and feedback, and specifically to: Bastiaan Beijer (KE-works), Anne-Liza Bruggeman (TU Delft), Jasper Bussemaker (DLR), Andrew Jeyaraj (Concordia University), Susan Liscouet-Hanke (Concordia University), Nikta Tabesh (Concordia University) and Bas van Manen (Fokker).

References

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

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

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

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

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

[6] NATO, "NATO Architecture Framework Version 4," 2018.

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

[8] S. Friedenthal, A. Moore and R. Steiner, A Practical Guide to SysML - The Systems Modeling Language, Waltham (US-MA): Elsevier, 2012.

[9] A. L. Ramos, J. V. Ferreira and J. Barceló, "Model-Based Systems Engineering: An Emerging Approach for Modern Systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 1, pp. 101-111, 2011.

[10] EC INEA Agency, AGILE 4.0 Project Consortium, "Grant Agreement Number 815122 - AGILE 4.0," 2019.

[11] P. D. Ciampa and B. Nagel, "Accelerating the Development of Complex Systems in Aeronautics via MBSE and MDAO: a Roadmap to Agility," in *AIAA Aviation Forum*, Washington (US-DC), 2021.

[12] P. D. Ciampa, L. Boggero, J. Vankan, T. Lefebvre, B. Beijer, A. Bruggeman, T. van der Laan, M. Fioriti, P. della Vecchia and J. B. Vos, "MBSE-MDAO agility in Aircraft Development: Achievements and Open Challenges from the AGILE4.0 Project," in *AIAA Aviation Forum*, Chicago (US-IL), 2022.

[13] L. Boggero, P. D. Ciampa and B. Nagel, "An MBSE Architectural Framework for the Agile Definition of System Stakeholders, Needs and Requirements," in *AIAA Aviation Forum*, Washington (US-DC), 2021.

[14] Department of Defense, "MIL-STD-499B - Systems Engineering," 1992.

[15] Electronics Industry Association, "EIA/IS 632 - Systems Engineering, Interim Standard," 1994.

[16] IEEE Computer Society, "IEEE 1220 - Standard for Application and Management of the Systems Engineering Process," 1998.

[17] International Organization for Standardization, "ISO/IEC 15288 - Systems and Software Engineering - Software Life Cycle Processes," 2002.

[18] INCOSE, Systems Engineering Handbook v.3, 2006.

[19] NASA, Systems Engineering Handbook Rev 2, 2016.

[20] S. A. Sheard and J. G. Lake, "Systems Engineering Standards and Models Compared," *INCOSE International Symposium - Vancouver (CA)*, vol. 8, no. 1, pp. 591-598, 1998.

[21] G. Chang, H. Perng and J. Juang, "A review of systems engineering standards and processes," *Journal of Biomechatronics Engineering*, vol. 1, no. 1, pp. 71-85, 2008.

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

[23] 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.

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

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

[26] 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.

[27] 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.

[28] 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.

[29] Object Management Group (OMG), "OMG Systems Modeling Language (OMG SysML™) - Version 1.4," 2015.

[30] 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.

[31] E. Baalbergen, J. Vankan, L. Boggero, J. Bussemaker, T. Lefebvre, B. Beijer, A. Bruggeman and M. Mandorino, "Advancing Cross-Organizational Collaboration in Aircraft Development," in *AIAA Aviation Forum*, Chicago (US-IL), 2022.

[32] T. van der Laan, "Bringing Manufacturing into the MDO domain using MBSE," in *AIAA Aviation Forum*, Chicago (US-IL), 2022.

[33] G. Donelli, P. D. Ciampa, N. Bartoli, T. Lefebvre, J. Mello, F. Odaguil and T. van der Laan, "Value-driven model-based optimization coupling design-manufacturing-supply chain in the early stages of aircraft development," in *AIAA Aviation Forum*, Chicago (US-IL), 2022.

[34] M. Fioriti, C. Cabaleiro, T. Lefebvre, P. della Vecchia, M. Mandorino, S. Liscouet-Hanke, A. Jeyaraj, G. Donelli and A. Jungo, "Multidisciplinary design of a more electric regional aircraft including certification constraints," in *AIAA Aviation Forum*, Chicago (US-IL), 2022.

[35] F. Torrigiani, S. Deinert, M. Fioriti, F. Di Fede, A. Jungo, L. Pisu, F. Sanchez, S. Liscouet-Hanke, P. D. Ciampa and B. Nagel, "MBSE Certification-Driven Design of a UAV MALE Configuration in the AGILE 4.0 Design Environment," in *AIAA Aviation Forum*, Washington (US-DC), 2021.

[36] M. Mandorino, P. della Vecchia, S. Corcione, F. Nicolosi, G. Cerino, M. Fioriti, C. Cabaleiro, T. Lefebvre, D. Charbonnier, Z. Wang and D. Peeters, "Multidisciplinary Design and Optimization of Regional Jet Retrofitting Activity," in *AIAA Aviation Forum*, Chicago (US-IL), 2022.

[37] J. Bussemaker, P. D. Ciampa, J. Singh, M. Fioriti, C. Cabaleiro, Z. Wang, D. Peeters, P. Hansmann, P. della Vecchia and M. Mandorino, "Collaborative Design of a Business Jet Family Using the AGILE 4.0 MBSE Environment," in *AIAA Aviation Forum*, Chicago (US-IL), 2022.