AIAA AVIATION Forum
August 2-6, 2021, VIRTUAL EVENT
AIAA AVIATION 2021 FORUM

10.2514/6.2021-3076

An MBSE Architectural Framework for the Agile Definition of System Stakeholders, Needs and Requirements

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

Model Based Systems Engineering (MBSE) approaches are rapidly spreading among organizations and industries due to all their claimed benefits over traditional document-based approaches. Benefits include for instance enhanced design quality of systems, clearer development of system requirements and specifications and improved communications within the design teams. Currently, MBSE methods and tools are mainly employed to successfully develop complex systems, such as aircraft or its components. However, this paper proposes to adopt MBSE also in the design of development systems, which aim to design complex systems. In particular, this paper focuses on the first activities of a typical Systems Engineering Product Development process: identification of system stakeholders, collection of their needs and development of system requirements. The main outcome delivered from this paper is an *architectural framework*, i.e. a guideline for the modeling of complex systems. More specifically, the architectural framework is still under development, and hence the current version focuses on the modeling of stakeholders, needs and requirements of complex systems. The focus of the proposed architectural framework is on the *agility* for the definition phases of complex systems. In other words, it is developed to streamline, improve and accelerate the definition and modeling of complex systems. Details of the architectural framework including the means to represent all the system information are provided. In addition, the architectural framework for the development of complex systems is supported by an MBSE development system, currently being addressed in the EU-funded research project AGILE 4.0. The MBSE development system is presented in this paper together with an example of its application for the definition of complex systems: an horizontal tail plane for a regional jet aircraft, designed and manufactured within an aeronautical supply chain consisting of different companies.

Nomenclature

ACRE	=	Approach to Context-based Requirements Engineering
INCOSE	=	International Council on Systems Engineering
MBSE	=	Model Based Systems Engineering
MDAO	=	Multidisciplinary Design Analysis and Optimization
OEM	=	Original Equipment Manufacturer
SysML	=	System Modeling Language

I. Introduction

THE process for designing a new aircraft has radically changed since the beginning of the aviation era until nowadays. The introduction of new technologies and the continuously increasing demand of higher performance are making the aircraft design process always more complex. This complexity is reflected in an

* 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.

increased number of designers and a larger amount of data and information produced. Design data address different aspects of the aircraft under development. For instance, they include requirements, specifications, descriptions and interfaces of the several systems, components and parts of the aircraft. In addition, data and information encompass organizational aspects of the design process, as design decisions, life-cycle of the project, which designers are involved. All the different elements composing the large amount of design data are linked together through different types of relationships. For example, design solutions derive from design decisions, which are taken according to system requirements. The modification of specific elements entails impacts on the rest of the design data and information. In addition, design data are authored by multiple people belonging to different engineering departments or organizations of the supply chain. This might hide many relationships among all the elements, therefore making difficult or even impossible the traceability among all these elements.

The organization of all the data produced during the design process can be guided through an *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 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]. In other words, an aircraft is a *system*, since all its *entities*, e.g. wings, fuselage, engines and on-board systems, are joined together to make possible the flight and transport of payload. Therefore, an *architecture* describes different aspects of the *system*. A *system 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. 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].

The guidelines recommended by *architectural frameworks* are traditionally followed by designers for the preparation of documents addressing the different aspects of the *system*. Requirements, specifications, technical descriptions are collected in textual documents or organized in tables. These documents also track the design decisions taken, showing the evolution of the *system* from the initial design to its realization. However, a document-based approach is affected by several disadvantages and limitations, for instance ambiguity, lack of clarity, misunderstandings, poor traceability. On the contrary, a model-based approach would make design activities easier, enhancing design quality, system specification and communications within the design team [8]. Models can be built and employed in place of documents to clearly represent multiple *system architectures*. 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, 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*.

MBSE practices and tools can be integrated in *development systems* supporting the development of complex aeronautical products. However, Ciampa *et al.* have demonstrated that MBSE approaches can be adopted also for the development of the *development systems* themselves, such as *Multidisciplinary Design Analysis and Optimization (MDAO) systems*, with the objective to accelerate their deployment and the operations, and in turn accelerate the development of the *complex systems* as well [10]. This novel approach, known as *AGILE Paradigm* [11], has been developed within the frame of the EU funded research project H2020 AGILE [12] coordinated by DLR, focusing on MDAO and hence targeting the development of *MDAO systems*. With the follow-up EU-H2020 AGILE 4.0 project, led again by DLR, the scope is enlarged to include all the main pillars of the aeronautical supply-chain: design, production, certification and manufacturing [13]. MBSE is leveraged to develop a *MBSE development system* for the modeling, assessment, and optimization of *complex systems* addressing the entire life cycle. Therefore, this *MBSE development system* extends the scope of the *MDAO system* developed in AGILE project by including all the upstream activities that anticipate the setup and deployment of MDAO processes, such as definition of *complex systems* (in terms of stakeholders, needs and requirements) and *system architecting* [14]. 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). Therefore, one of the tasks of the AGILE 4.0 project is to leverage MBSE practices to develop *MBSE development systems* and to define an *architectural framework* to guide the development of *complex systems*.

The present paper focuses on this first step of any Systems Engineering process. It is indeed important to correctly identify and involve all the stakeholders, clearly understand what they expect from the system, and develop

complete, unambiguous and clear requirements. In fact, a report published by the Standish Group [15] claims that 13.1% of the project failures is due to an incomplete list of requirements, 12.4% is caused by lack of stakeholders involvement, 9.9% is attributed to unrealistic expectations and 8.7% is related to changing of requirements during the development process. In other words, almost half of the project failures can be mitigated or avoided if solutions would be taken to assist the designers during this first step of the process. The aim of this paper is therefore to develop an *architectural framework* that would guide the designer in the first step of the Systems Engineering Product Development process. Many guidelines identified from the literature review are included in the proposed *architectural framework*, such as the usage of rules, attributes and patterns. More specifically, many rules and attributes recommended by the International Council on Systems Engineering (INCOSE) [16] are included. A model-based approach based on SysML is adopted, although requirement statements are still expressed in natural language, in order to entail the correct interpretation from all the involved stakeholders. It is also recognized that the standard SysML profile is not effective to model requirements including many of the attributes recommended by INCOSE. Therefore, an extension of the profile is proposed. The *architectural framework* described in this paper is in fact being developed within the context of AGILE 4.0 project.

In order to reach the previously stated objectives, this paper is organized as follows. After the introduction of Section I, an overview of other main studies found in literature is presented in Section II. These studies represent the starting point for the definition of the *architectural framework* proposed in this paper, which is presented and deeply described in Section III. The *MBSE development system* developed to support the *agile* definition and modeling of system stakeholders, needs and requirements is addressed in Section IV. An example of application of the architectural framework focusing on a specific *complex system* that can be designed through *the MBSE development system* is provided in Section V. The paper ends with Section VI by deriving conclusions and suggesting future developments.

II. Literature review on the definition and modeling of stakeholders, needs and requirements

Many *Systems Engineering Product Development processes* have been proposed until nowadays for the development of *complex systems*, part of them in the form of standard (e.g. [17], [18], [19] and [20]), others described in Systems Engineering handbooks (e.g. [21] and [22]). The interested reader can find an overview of the main Systems Engineering Product Development processes in [23] and [24]. All these processes start with the identification of system stakeholders, collection of their needs and development of system requirements. These activities of a Product Development process are the focus of the present paper.

Several research studies that aim to improve the first step of a Systems Engineering Product Development process propose solutions focusing on the development of requirements. For instance, Génova *et al.* [25] identify desirable properties that textual requirements should meet. Indeed, requirement statements should be verifiable, complete, consistent, understandable, unambiguous and traceable. Therefore, measurable indicators are proposed by the authors to subjectively judge the quality of requirements, i.e. how well requirements meet the desirable characteristics. A similar list of characteristics that textual requirements should have is suggested by the INCOSE in its guide for writing requirements [16]. In addition, INCOSE recommends 44 rules that if followed would assure that textual requirements are compliant with the quality characteristics. An example of rule is the adoption of a pre-defined structure for requirement statements. This structure is named pattern, and it makes textual requirements complete, consistent, comprehensible and able to be validated. In other words, the pattern prescribes which elements should be included in the requirement statement, e.g. subject, performance characteristics, and conditions. Several sources propose patterns for the writing of textual requirements, e.g. [26]. An interesting work dealing with textual requirement patterns is proposed in [27], where the author suggests a few kinds of pattern according to the type of requirement. However, only writing high quality requirement statements is not sufficient to assure the success of the project. Therefore, INCOSE [16] and other authors (e.g. [28], [29]) propose several attributes for the management of requirements. Examples of attributes are: requirements' author, sources from which requirements are derived, means of compliance. In addition, a structured approach can guide the designer in representing all the information related to requirements. An example of this structured approach is the *ACRE (Approach to Context-based Requirements Engineering)*, developed by Holt *et al.* [30]. The authors propose an architectural framework for the organization of requirement data in different *views*. **A *view* is a representation of part of the system model, with a determined focus**. For example, a view of ACRE might contain all the rules that requirements should follow, while another view can describe how each requirement can be validated. All this information can be documented in reports, tables or databases, according to a document-based approach. However, a lot of effort is being made in literature to model the requirements due to all the potential advantages previously mentioned. Several standard modeling languages are used by the research community in MBSE contexts, e.g. (IDEF0) [31], OPM [32] AADL [33] and UPDM [34].

However, the majority of the studies adopt the Systems Modeling Language (OMG SysML) [35], since it is recommended by INCOSE for MBSE activities [21]. SysML extends the Unified Modeling Language (UML) profile [36] with new diagrams and stereotypes, including those for the modeling of requirements. Indeed, standard *SysML Requirement Diagrams* are employed to represent requirement statements with some attributes (e.g. requirement ID) and to link requirements to other elements of the system model, e.g. through *derivation* and *verification* relationships. Nevertheless, requirement statements in SysML are expressed in natural language, which can entail drawbacks as ambiguity and miscommunication. In order to avoid this, innovative approaches are proposed (e.g. [37], [38]), where SysML elements are used to model requirement statements. However, other researchers (e.g. [39]) suggest the definition of requirements in natural language, so they can be easily communicated to all the stakeholders.

All the mentioned studies have been used as reference in the present paper to develop the research activities addressed in the following Sections.

III. A model-based architectural framework for the development definition of stakeholders, needs and requirements

As already introduced earlier, the paper focuses on the first step of a Systems Engineering Product Development process, which aims at identifying system stakeholders, collecting their needs and deriving system requirements. A ***stakeholder*** is defined as an "***individual or organization having a right, share, claim, or interest in a system or in its possession of characteristics that meet their needs and expectations***" [20]. Given an aircraft as a *system*, examples of stakeholders encompass: airlines, passengers, maintainers, Original Equipment Manufacturers (OEMs), suppliers, regulation authorities, ground crews, cabin crews and air traffic controllers. All the stakeholders expect from the system different wishes, necessities and desires. In other words, they express different ***needs***, i.e. "***informal expressions of something that has to be provided, ensured or avoided by a system or the development project of this system***" [26]. For example, airlines might want to maximize profit, while passengers would demand a safe and comfortable flight. Since *needs* are generally unstructured and expressed in fuzzy or general, ambiguous terms, they must be used to originate *requirements*, which on the contrary should follow precise patterns and rules to assure characteristics as unambiguity, completeness, feasibility, verifiability and correctness. A ***requirement*** is defined as "***a statement which translates or expresses a need and its associated constraints and conditions***" [26].

The *architectural framework* described in this Section and schematically represented in Figure 1 is developed to guide the designer during the process for the definition of stakeholders, needs and requirements.

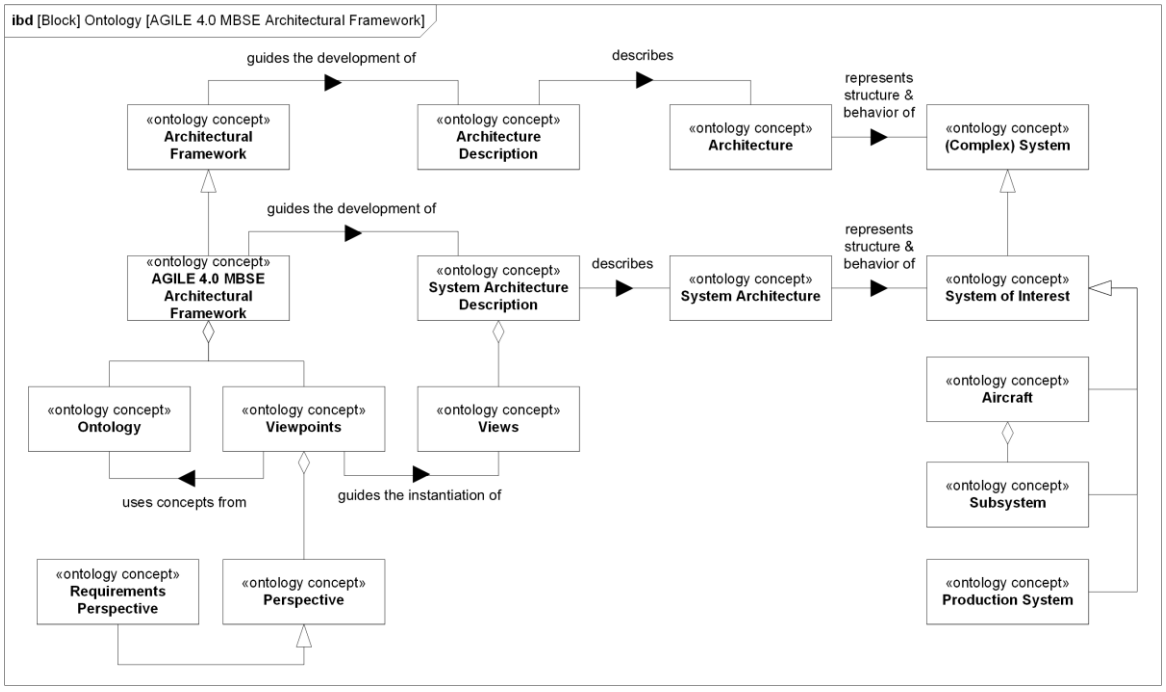

Figure 1 Ontological representation of the AGILE 4.0 MBSE architectural framework representing key concepts of the framework and their relationships.

The AGILE 4.0 MBSE architectural framework guides the development of architecture descriptions of different types of aeronautical System of Interest, as aircraft and related subsystems (e.g. wing, engine, landing gear system) and production systems (e.g. manufacturing system, supply chain system). **As any *architectural framework*, the one being developed by the project Consortium is composed by *ontologies* and *viewpoints***, as explained in [40]. **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 descriptions* from the perspective of specific system concerns, named *perspectives***. ***Viewpoints* are prescribed to create *views* for the description of the *architectures* of the system under development from the different *perspectives***. In this specific case, a single perspective is identified, which is the *Requirements Perspective*. The remaining part of the present Section will describe the *ontology* and the *viewpoints* of the proposed *architectural framework* for the representation of stakeholders, needs and requirements of *complex systems*.

A. Architectural framework: *ontology* for stakeholders, needs and requirements

The *ontology* depicted in Figure 2 aims at defining the key concepts previously introduced and their relationships: complex system, MBSE development system, stakeholders, needs, requirements. In addition, the *ontology* includes other concepts that are explained below, as requirement types, rules and attributes. Another example of *ontology* is provided by Holt *et al.* [30], but it doesn't cover all the key concepts identified in this paper and previously listed. Another example is proposed in [29], even if it is not referred as an *ontology* although showing key concepts and relationships between them. From this example, the *ontology* proposed in this paper is realized, with some modifications.

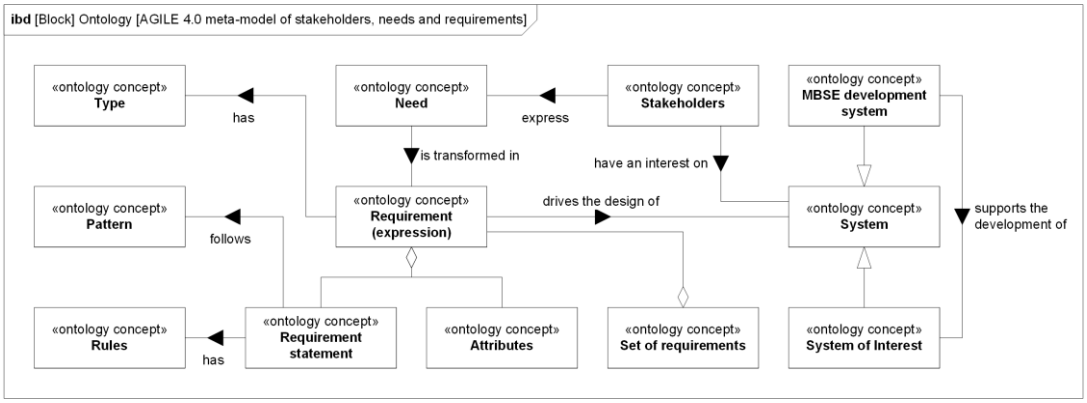

Figure 2 Ontology included in the proposed architectural framework representing key concepts and their relationships for the definition of system stakeholders, needs and requirements.

This *ontology*, developed within the context of the AGILE 4.0 research project, has been published as open access [41]. Therefore, it can be freely downloaded from the Community page of the project on the Zenodo website [42] and re-used by any user inside and outside the project Consortium. The available files represent the meta-models, rendered by OWL, supporting the development of any complex system in any domain.

One of the key concepts of the *ontology* is the *system*, as defined in the introductive Section I. Within the specific context of the present paper, the system can be a *complex system* (e.g. an aircraft), or an *MBSE development system*, which aims at supporting the designer in the development of the *complex system* (e.g. through tools and file formats). The first task of the Systems Engineering Product Development process targets the identification of all the *stakeholders* that have an interest on the system. All the stakeholders express *needs*, which are then transformed into *requirement expressions*, or simply *requirements*. The system must be designed according to requirements. Multiple requirements can be grouped together, for instance when dealing with the same subject. Groups of requirements are named *sets of requirements*. Each requirement is formed by two parts: the former is the *requirement statement*, i.e. the text. The latter is a series of *attributes* that are used for the management of the requirement. Several *attributes* are recommended and explained in [16], and examples encompass: requirement ID, author and means of compliance. Requirements can also be categorized according to their *type*. For instance, *functional requirements* define which functions should be performed by the system, while *performance requirements* specify how well these functions should be performed [22]. According to the type, requirement statements follow different *patterns*, i.e. they contain mandatory and optional elements including for instance functions, performance characteristics, durations and conditions. Finally, requirement statements should comply several *rules*, for example those

recommended by INCOSE [16], to assure all the quality characteristics introduced in Section I (e.g. unambiguity, completeness, verifiability).

More details about the requirement types, patterns and attributes adopted in the proposed *architectural framework* are provided in the description of the *viewpoints*, in the following subsection.

B. Architectural framework: *viewpoints* for stakeholders, needs and requirements

Ten different novel *viewpoints* are proposed for the representation of stakeholders, needs and requirements. All these *viewpoints* belong to the *requirement perspective*, but each one of them focuses on specific aspects of the system. The *SysML Package Diagram* depicted in Figure 3 collects all the *viewpoints*.

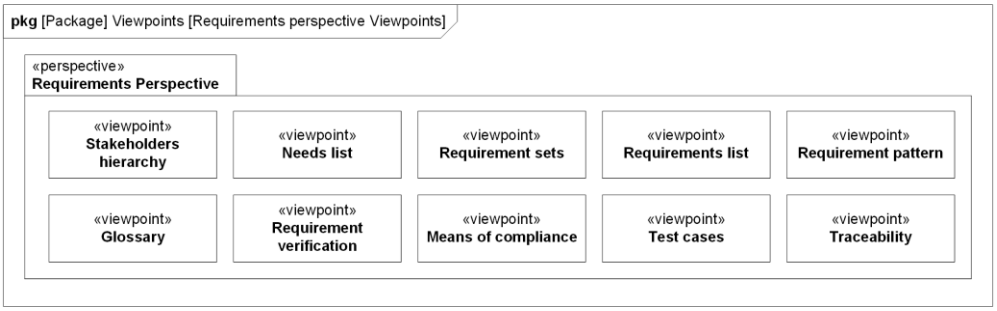

Figure 3 *Viewpoints* of the proposed *architectural framework* belonging to the *requirement perspective*. A *SysML Package Diagram* is adopted to model the *viewpoints*.

The following subsections present the modeling guidelines prescribed by the *viewpoints* recommended in the *architectural framework*. A modified profile derived from SysML has been created to represent all the elements (e.g. stakeholders and needs), requirements attributes and patterns previously explained. This new profile is named *AGILE4Profile*, and it is described in the *Appendix*.

1. Viewpoints "Stakeholders hierarchy" and "Needs list"

Two different *viewpoints* are proposed to represent the model of system stakeholders and their needs. The *viewpoint* "Stakeholders hierarchy" provides guidelines to represent all the stakeholders involved with the system under development during the whole life-cycle of the system. Figure 4 (a) illustrates the template of the *views* that can be obtained according to this *viewpoint*. The new stereotype *«systemStakeholder»* defined in the *AGILE4Profile* is used to represent all the stakeholders. Moreover, it can be noted that the present *viewpoint* can be used to depict the multi-lever hierarchy among stakeholders.

A *SysML Package Diagram* like the one of Figure 4 (b) can instead be employed to represent all the needs collected from the different system stakeholders. Also for this *viewpoint*, a new stereotype is derived from SysML. This new element of the *AGILE4Profile* is named *need*, and other than reporting each need text and ID, it also specifies as information who is the stakeholder who owns the specific need.

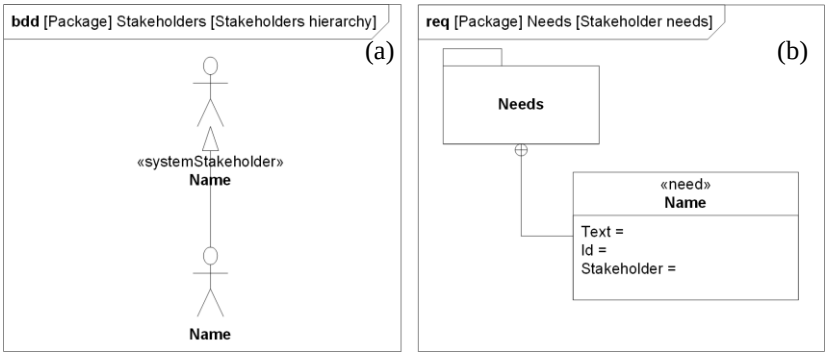

Figure 4 (a) *SysML Block Definition Diagram* representing the template prescribed by the *viewpoint* "Stakeholders hierarchy" to model all the system stakeholders and the hierarchy among them. (b) *SysML Requirement Diagram* representing the template prescribed by the *viewpoint* "Needs list" to collect all the stakeholder needs.

2. Viewpoints "Requirement sets", "Requirements list", "Glossary" and "Traceability"

Next viewpoints aim at collecting all the project requirements and relate them among each other and with needs. More specifically, the *viewpoint* "Requirement sets" aims at illustrating all the identified sets according to which requirements are grouped. Aim of the *viewpoint* "Requirement lists" is to represent the system requirements. As represented in Figure 5 (a), the *viewpoint* "Requirement sets" prescribes the realization of a *SysML Package Diagram*, where different *package* elements are used to define each set.

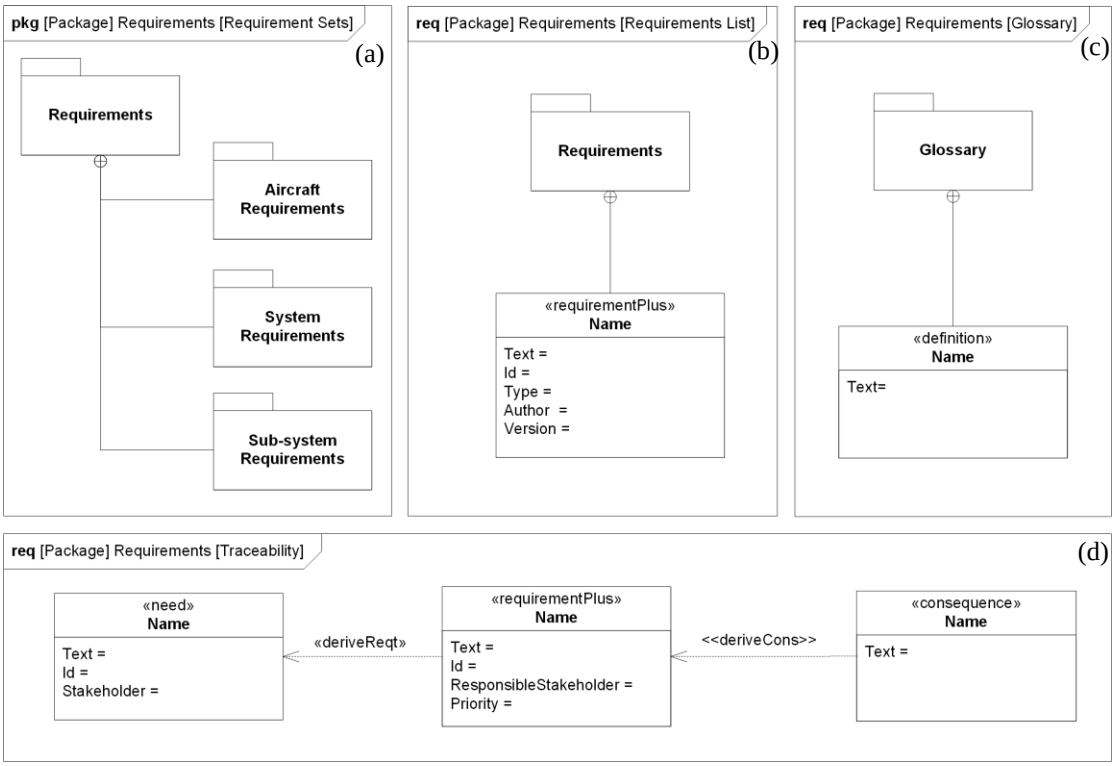

Figure 5 (a) *SysML Package Diagram* representing the template prescribed by the *viewpoint* "Requirement sets" to depict all the requirement sets. (b) *SysML Requirement Diagram* representing the template prescribed by the *viewpoint* "Requirements list" to collect all the requirements belonging to each requirement set. (c) *SysML Requirement Diagram* representing the template prescribed by the *viewpoint* "Glossary" to collect definitions and nomenclature. (d) *SysML Requirement Diagram* representing the template prescribed by the *viewpoint* "Traceability" to show the derivation relationships among needs, requirements and consequences.

A *SysML Requirement Diagram* is instead selected for the modelling of the requirements list. Figure 5 (b) shows a template of diagram that should be adopted in the creation of *views* compliant with this *viewpoint*. The *requirementPlus* element is part of the *AGILE4Profile*, and it is connected through a *containment* relationship to a *package* element representing the whole *requirements set*. Different *views* can be created according to this *viewpoint*, each one focusing on a *requirements set*. This *viewpoint* should include the following properties per each requirement:

- **Text**, i.e. the *requirement statement*, by following the prescribed *patterns* according to the requirement *type*. This property is part of the metaclass «*requirement*» of the standard SysML profile.
- **ID**, i.e. a unique identifier of the requirement, which can be either a number or a mix of numbers and characters, needed to refer to the specific requirement, as recommended by INCOSE [16]. Same as the *text*, this property is part of the metaclass «*requirement*» of the standard SysML profile.
- **Type**, e.g. functional or performance requirement.
- **Author**, including name and optionally affiliation (i.e. department or company) of requirement manager.
- **Version**, which can be updated after changes of the requirement.

Requirement statements and needs might also contain very specific terminology and abbreviation, for which definitions should be provided. The *viewpoint* "Glossary" is therefore conceived to collect and explain all the

nomenclature used in the model. The *SysML Requirement Diagram* depicted in Figure 5 (c) shows the template prescribed by this *viewpoint* for this specific purpose. The *viewpoint* is characterized by the new stereotype *«definition»*, which is introduced with the *AGILE4Profile* and reports the explanations of each terminology and abbreviation.

Traceability is one of the main advantages claimed by a model-based approach. All the elements characterizing the system under design, in this paper mainly stakeholders, needs and requirements, are connected together through a relationship, which can be for example of the type *derivation*. The *views* realized by following the guidelines provided by the *viewpoint* "Traceability" show which requirements are derived from which needs and which are the consequences in case the developed system is not compliant with requirements. The *SysML Requirement Diagram* of Figure 5 (d) represents the template prescribed by this *viewpoint*. The two elements *need* and *requirementPlus* are linked by a *deriveReqt* relation, which belongs to the standard SysML profile. As explained before, the *need* element is introduced with the *AGILE4Profile*, and in addition to the need's text and identifier, it shows the stakeholder who expresses the specific need. The *requirementPlus* element reports the text and identifier of the requirement. Moreover, it reports the stakeholder who is responsible of the verification of the requirement, while the attribute *priority* specifies how much the requirement is important to stakeholders [16]. Multiple requirements can be connected together through the *deriveReqt* relation, but requirements can also be linked to *consequence* elements through the *deriveCons* relation. Both the stereotypes *«consequence»* and *«deriveCons»* are extensions from the SysML.

3. Viewpoint "Requirement pattern"

As explained in Section II, requirement statements have to be complete, consistent, comprehensible and able to be validated, otherwise the project might not be successful. Therefore, five types of pattern are prescribed by this *viewpoint* to model requirement statements by including in the text all the necessary elements, e.g. subject, functions, and conditions.

Each requirement pattern depends on the type of requirement, as reported below:

- **Functional requirements**: define what functions need to be performed to accomplish the objectives [22]
  Pattern: The SYSTEM shall [exhibit] FUNCTION [while in CONDITION]
  Example: "The aircraft shall provide propulsive power [during the entire mission]"

- **Performance requirements**: define how well the system needs to perform the functions [22]
  Pattern: The SYSTEM shall FUNCTION with PERFORMANCE [and TIMING upon EVENT TRIGGER] while in CONDITION
  Example: "The aircraft shall fly at min Mach 0.8 during cruise"

- **Design constraint requirements**: limit the options open to a designer of a solution by imposing immovable boundaries and limits [27]
  Pattern: The SYSTEM shall [exhibit] DESIGN CONSTRAINTS [in accordance with PERFORMANCE while in CONDITION]
  Example: "The aircraft shall have technologies with maturity TRL 9"

- **Environmental requirements**: define which characteristics the system should exhibit when exposed in specific environments (e.g. acoustic/thermal loads, atmospheric conditions) [27]
  Pattern: The SYSTEM shall [exhibit] CHARACTERISTIC during/after exposure to ENVIRONMENT [for EXPOSURE DURATION]
  Example: "The aircraft shall be maneuverable during exposure to ice conditions [for the entire flight]"

- **Suitability requirements**: include a number of the "-ilities" in requirements to include, e.g. transportability, survivability, flexibility, portability, reusability, reliability, maintainability, and security [27]
  Pattern: The SYSTEM shall exhibit CHARACTERISTIC with PERFORMANCE while CONDITION [for CONDITION DURATION]
  Example: "The aircraft shall exhibit a steady gradient of climb of minimum 2.4% while condition of one-engine-inoperative"

According to this *viewpoint*, each requirement statement can be modeled in compliancy to its pattern and represented in a *SysML Requirement Diagram*. Figure 6 shows one of the five types of diagram that can be realized

according to this *viewpoint*. This specific diagram is a template that can be employed to represent *performance requirements*, and it includes all the elements that are optional or mandatory in the statement. New *stereotypes* are derived from SysML and part of the *AGILE4Profile*, namely *«system»*, *«function»*, *«performance»*, *«timing»*, *«eventTrigger»* and *«condition»*. Additional *stereotypes* can be included in *views* conforming with the *viewpoint* "Requirement pattern", i.e. *«designConstraints»*, *«environment»*, *«exposureDuration»*, *«characteristic»* and *«conditionDuration»* (more details can be found in the *Appendix*). Moreover, it can be noted that the *requirementPlus* element illustrated in this viewpoint should also include the property *Syntax Verification*, which is a requirement attribute stating the compliancy status of the requirement statement with all the rules that assure all the quality characteristics mentioned in Section I, as unambiguity, completeness and verifiability.

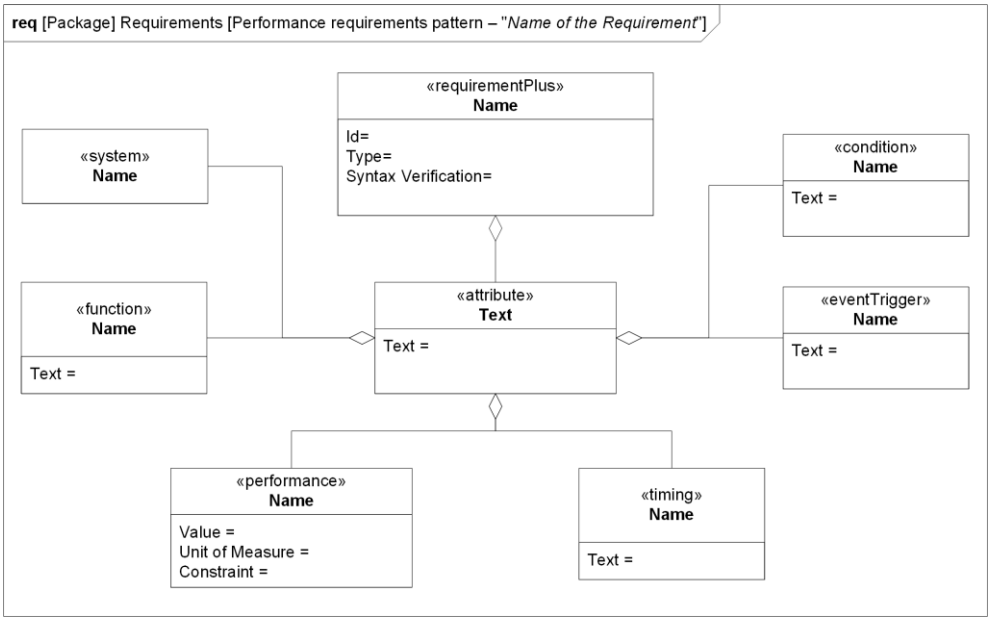

Figure 6 *SysML Requirement Diagram* representing the template prescribed by the *viewpoint* "Requirement pattern" to depict performance requirement statements. Similar diagrams but with different *stereotypes* (details in the *Appendix*) can be used to represent the requirement statements of the other requirement types.

4. Viewpoints "Means of Compliance", "Test Case" and "Requirement verification"

The last three *viewpoints* addressed by the *AGILE 4.0 architectural framework* prescribe guidelines for the representation of how requirements are verified, i.e. how the designer can prove that the system under design is going to be compliant with the collected requirements. Two main types of element should be identified for verification purposes: *Means of Compliance* and *Test Cases*. *Means of Compliance* have been defined by the project Consortium as generic ways to prove the compliancy of the system with requirements. Examples of Means of Compliance include "simulations", "tests" and "disciplinary analyses". *Test Cases* are instead instantiation of the Means of Compliance. In other words, Test Cases are specific ways exploited for requirements verification, and they can encompass for example specific software – which can belong to the Means of Compliance: disciplinary analysis – and experiments carried out with pre-determined procedures and equipment, which can belong to the Means of Compliance: test.

The *SysML Package Diagram* of Figure 7 (a) is prescribed by the *viewpoint* "Means of Compliance" to collect all the Test Cases that belong to each Means of Compliance. Every Test Case is represented by the new element of the *AGILE4Profile* named *testCasePlus*, which should include as property the identification code ID of the Test Case and the name of the diagram describing how the Test Case functions. Indeed, a different *viewpoint* named "Test Case" prescribes the usage of *SysML Sequence Diagrams* as illustrated in Figure 7 (b) to represent how each Test Case (represented by a *block* element) functions and is operated by the designers, which are in charge of verifying each requirement.

Finally, Means of Compliance and Test Cases can be selected and associated to requirements for their verification. This information is provided through views compliant with the *viewpoint* "Requirement Verification",

which consist of *SysML Requirement Diagrams* like the one of Figure 7 (c). Diagrams compliant with this *viewpoint* include the *requirementPlus* element, which other than the requirement statements, reports the following properties:

- **System Verification**, to state if the system has already been verified against the requirement, and the outcome obtained from the verification process.
- **Validation**, to state whether the requirements has been correctly derived from other requirements or needs.
- **Means of Compliance**, i.e. a generic way to verify the requirement
- **Responsible Stakeholder**, who is the person or group of people in charge of designing the system as prescribed by the requirements.

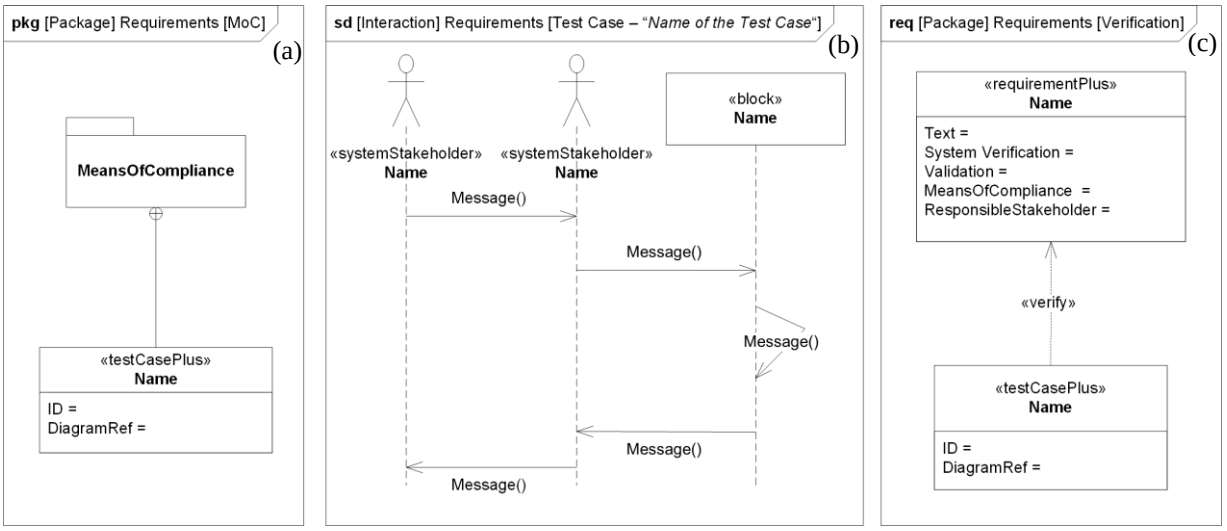

Figure 7 (a) *SysML Package Diagram* representing the template prescribed by the *viewpoint* "Means of Compliance" to collect all the Test Cases divide per Means of Compliance. (b) *SysML Sequence Diagram* representing the template prescribed by the *viewpoint* "Test Case" to describe the functioning of each Test Cases and its operations by designers. (c) *SysML Requirement Diagram* representing the template prescribed by the *viewpoint* "Requirements Verification" to show how each requirement is verified.

IV. The development of the MBSE development system through an MBSE approach

An MBSE approach is adopted to setup the *MBSE development system* supporting the *architectural framework* addressed in Section III. This MBSE approach starts with the modeling of the stakeholders, needs and requirements of the *MBSE development system*. The *SysML Requirement Diagram* of Figure 8 reports the functional requirements of the *development system*. More specifically, these requirements have been derived by the needs expressed by an industrial partner of the project Consortium, namely Bombardier Aerospace (BA).

All the collected functional requirements represented in the *view* of Figure 8 address functionalities that the *MBSE development system* shall have in order to be *agile*, i.e. to streamline, improve and accelerate the definition and modeling of complex systems. Some requirements specify what should be addressed by the *development system*, namely the definition of (aeronautical) system stakeholders, needs and requirements. More precisely, the definition of these elements should occur according to a model-based approach, as stated by requirement with ID: MBSE.04. The *MBSE development system* shall also be the Single Source of Truth of the modeled information, and it should allow multiple partners to perform all the design tasks in a collaborative way. More levels of system abstraction should be addressed by means of the *development system*, e.g. aircraft-level, subsystem-level and component-level. Moreover, technologies should be included for automatic activities as verification of the correctness and completeness of the model and generation of requirement statements. The traceability among all the model elements should also be fostered by the *MBSE development system*, and all the generated information should be collected and represented in *views* conforming with the *viewpoints* described in Section III. Finally, the *development system* shall support the re-use of the model (or part of it) in follow-up projects.

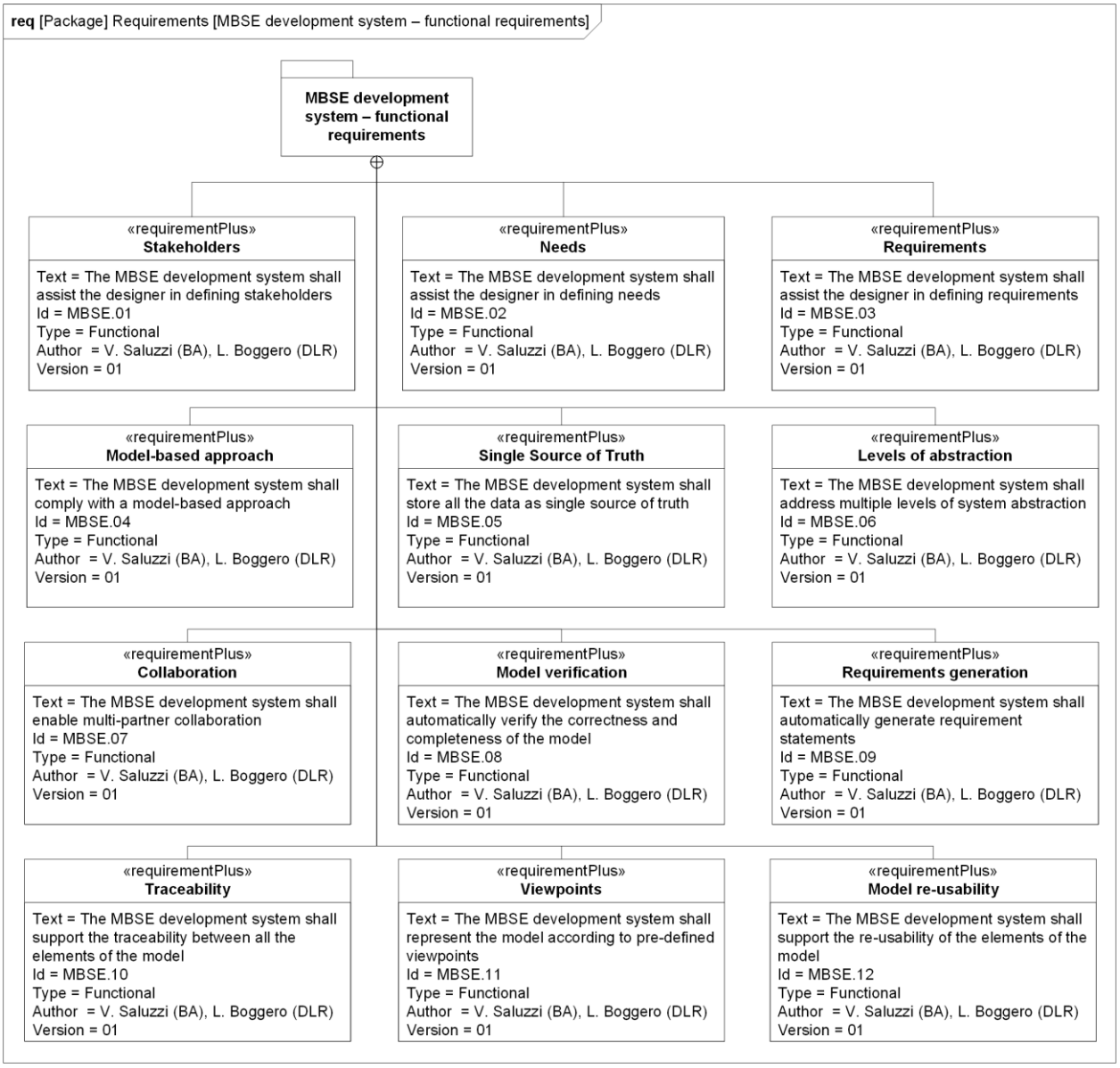

Figure 8 *SysML Requirement Diagram* representing the *view* collecting the functional requirements of the *MBSE development system*. These functional requirements have been derived the needs expressed by an industrial partner of the project Consortium, i.e. Bombardier Aerospace (BA).

Then, a *logic architecture* of the *MBSE development system* is derived, by identifying all the logic components that are needed to fulfill the functionalities expected from the *development system*. The logic architecture is indeed solution-independent, meaning that it doesn't specify any tools or software that are part of the *MBSE development system*. This task is in fact addressed by another step of the MBSE process, in which specific tools and software are identified and mapped to the logic components previously identified, entailing therefore the so-called *physical architecture*. Figure 9 depicts the logic architecture of the *MBSE development system* through a *SysML Internal Block Diagram*. Four logic components are required in order to collect as input from the designer all the data relative to system stakeholders, needs and requirements (including attributes). In particular, requirement elements – e.g. functions, performance characteristics and conditions – should be used by one of the logic components to automatically generate requirement statements. All the collected and produced data should be therefore stored in a single place, in compliancy with the functionally relative to the Single Source of Truth characteristic demanded by the functional requirement MBSE.05. Moreover, a *traceability management* logic component should be designed and integrated in order to easily handle all the relation between the different elements of the model. As prescribed by

one of the required functionalities, a logic component should be included into the *MBSE development system* to verify that all the data is correct and complete, and report the verification results to the designer. Finally, all the collected and produced data has to be represented in different *views*, consisting of the SysML diagrams prescribed by the *viewpoints* of the *AGILE 4.0 architectural framework*, and shown to the designer in a *model visualization* component.

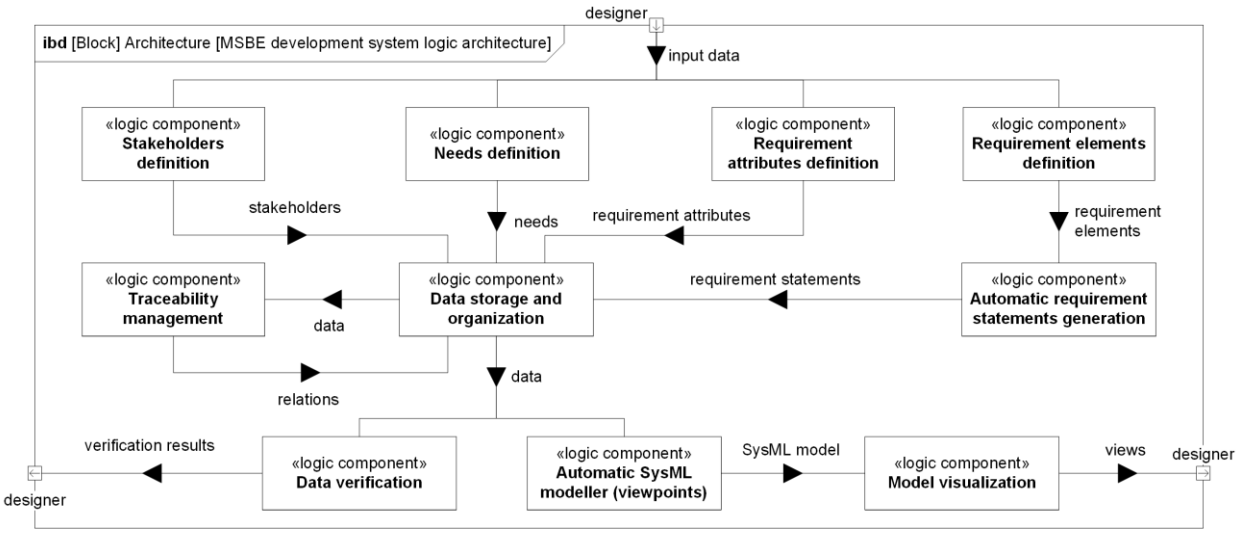

Figure 9 *SysML Internal Block Diagram* representing the logical architecture of the *MBSE development system*. This *view* illustrates the solution-independent components that should be integrated into the *development system* to fulfilled the required functionalities.

Finally, the physical architecture of the *MSBE development system* is designed and realized, as represented in Figure 10.

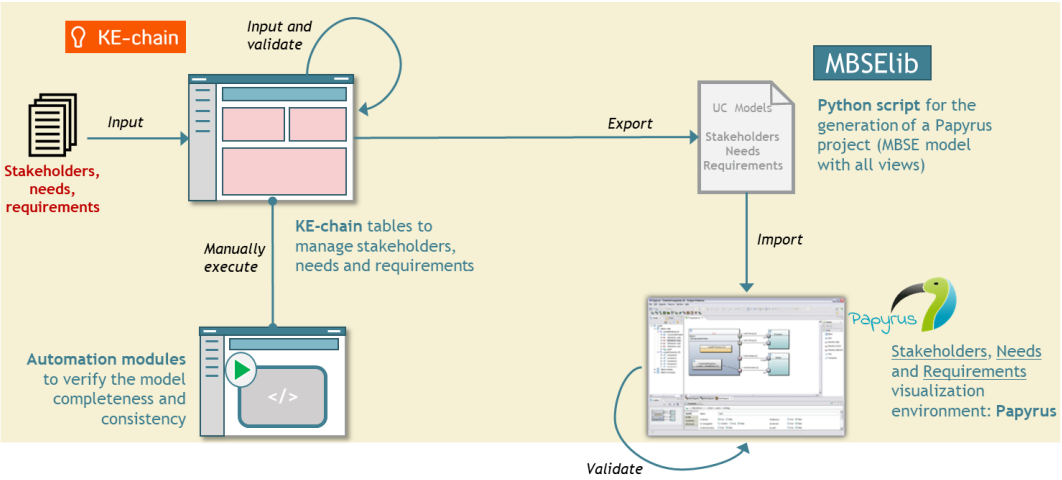

Figure 10 *View* representing the physical architecture of the *MBSE development system*. This *view* shows which software is integrated into the *development system* according to the logic components defined in the logical architecture.

The physical architecture of the *MSBE development system* is realized by identifying and integrating existing software or by developing new tools, in order to instantiate all the components identified with the logic architecting activity. More specifically, the engineering platform KE-chain[§] provided by the AGILE 4.0 project partner KE-works serves as front end of the *MBSE development system* and as a process modeler. Multiple users can access the platform and specify through tables and interfaces, all the data of the system under development, as stakeholders,

[§] https://ke-chain.com/

needs, requirement elements and requirement attributes. Moreover, KE-chain can be exploited to define all the connections between the different elements, therefore addressing the functionality of supporting the traceability of the model. All the collected data can be then verified to assure completeness and correctness qualities. A python script is therefore integrated into the *MBSE development system* for this purpose, and after the execution, the obtained verification results are displayed through a dedicated interface in KE-chain. Once the model is created from all the elements provided by the designers and the relations between them, it can be exported and visualized in the form of SysML diagrams, according to the *viewpoints* prescribed by the *AGILE 4.0 architectural framework*. This happens automatically thanks to the DLR's tool MBSElib, which has been conceived to generate all the system *views* and save them as a Papyrus[**] project file. This automatically created file includes the models of stakeholders, needs and requirements, ant it can be opened in the Papyrus environment for inspection and verification by the designers.

V. Application of the MBSE development system for the modeling of stakeholders, needs and requirements of complex systems

As mentioned in the Introduction in Section I, the *MBSE development system* being addressed in AGILE 4.0 project aims to support the designers in developing *complex systems* through an *architectural framework*. The objective of present Section is to show an example of application of the *AGILE 4.0 MBSE development system* for the initial design of two aeronautical *systems*: a regional jet aircraft and its horizontal tail plane. This example of application is selected from AGILE 4.0, and it focuses on the design and manufacturing stages of the two systems. The interested reader can find more information about this application study in [43]. Another example of application of the *architectural framework* and *development system* for the design and certification of an Unmanned Aerial Vehicle can instead be found in [44], also this one addressed within the frame of the AGILE 4.0 project.

The whole aircraft of the application case shown in this paper and all its parts – including the horizontal tail plane – are designed, produced and assembled by a supply chain consisting of multiple aeronautical companies. The supply chain for the development and manufacturing of the regional jet aircraft and the horizontal tail plane is illustrated in the "Stakeholders" *view* of Figure 11, represented in the Papyrus environment integrated into the *MBSE development system*. In particular, this *view* shows a partial list of stakeholders that have a stake with the two systems during their life-cycle, e.g. maintainers, airlines, ground and fight crews and passengers. Among these stakeholders, some of them are part of the aircraft supply chain. The ***OEM*** is in charge of designing the whole aircraft and outsourcing the development and production of components to ***Tier I suppliers***. In this specific application case, Tier I suppliers design and produce the horizontal tail plane, but they can outsource the production of smaller components (e.g. structural parts) to different ***Tier II suppliers***. Once the horizontal tail plan is developed and produced, it is integrated by the OEM into the whole aircraft.

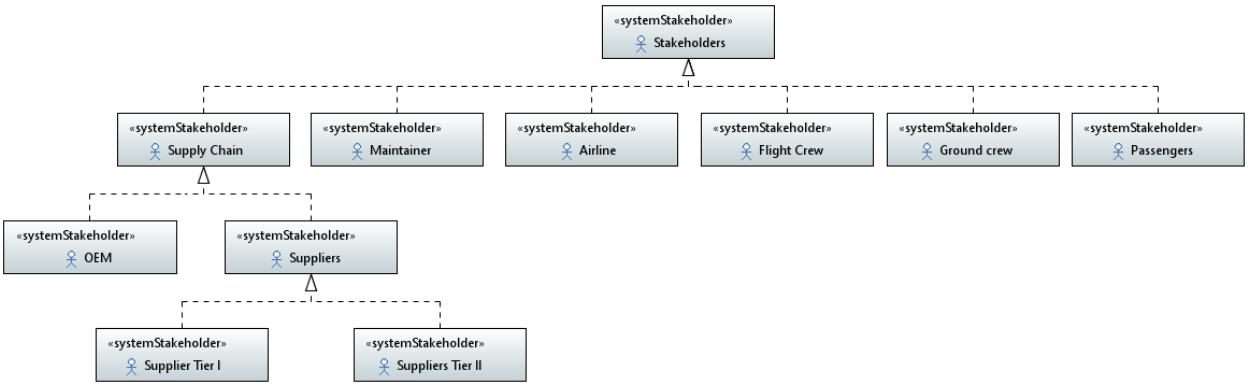

Figure 11 *SysML Block Definition Diagram* representing the "Stakeholders" *view* in Papyrus showing some stakeholders of the two systems, i.e. the regional aircraft and the horizontal tail plane [43].

As already explained before, all the stakeholders express different needs. In this example, some needs of the OEM are modeled and represented in the "Needs" *view* of Figure 12. The relative *viewpoint* explained in Section III is followed and the relative *SysML Requirement Diagram* is automatically obtained through the *MBSE development*

[**] https://www.eclipse.org/papyrus/download.html

system. Moreover, it should be noted that some of the needs (e.g. N-0024) are addressing the entire aircraft, while others (e.g. N-0052) are addressing parts of the regional jet, as the horizontal tail plane.

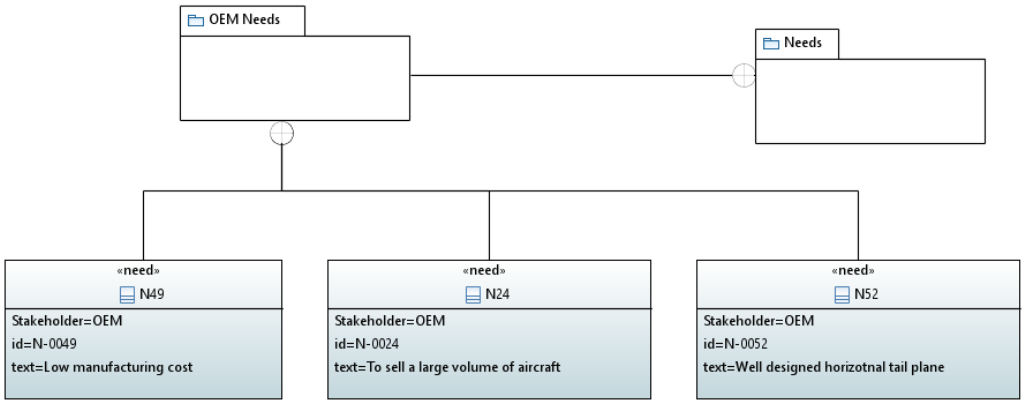

Figure 12 *SysML Requirement Diagram* representing the "Needs" *view* in Papyrus showing some of the OEM's needs.

From the view of Figure 12 it can also be noted that the OEM needs are quite vague. For example, it should be clarified what does "*well designed*" horizontal tail plane addressed in need N-0052 mean to the OEM. Indeed, all the needs should be transformed into requirements, following all the guidelines explained in Section III, and therefore assuring qualities as completeness, consistence and unambiguity.

Figure 13 shows the "Traceability" view, from which it can be seen how needs are refined by derived requirements.

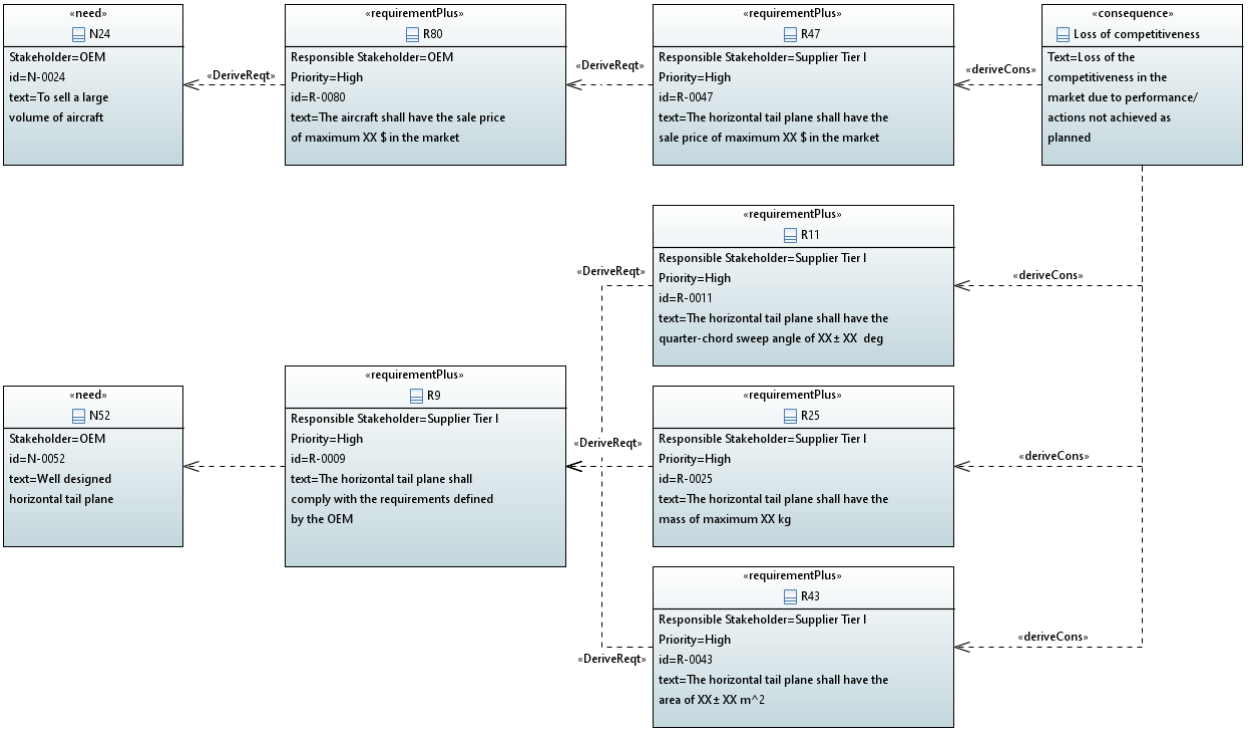

Figure 13 *SysML Requirement Diagram* representing the "Traceability" *view* in Papyrus showing the relationships between some needs, derived requirement at aircraft and horizontal tail plane levels and derived consequences [43].

More specifically, the OEM's need relative to the large volume of the aircraft to be sold generates a requirement dealing with the competitiveness of the whole aircraft, which characterizes the maximum price of the regional jet. From this aircraft-level requirement, a new one can be derived, this time addressing the maximum price of the horizontal tail plane. In addition, the two requirements have two different responsible stakeholders, since the OEM is in charge of developing an aircraft whose price is not higher than what stated in R-0080, and the Tier I supplier has to produce a component compliant with the sale price defined in R-0047. Other derived requirements refine the vague need previously mentioned about the "*well designed*" horizontal tail plane expected by the OEM. This need is indeed transformed into clear and unambiguous requirements, prescribing the geometrical and physical characteristics that the component should have in order to fulfill the OEM's expectations. These requirements are defined by the OEM, but it is the Tier I supplier who is responsible in developing and producing a tail plane compliant with them.

The diagram of Figure 13 shows also which are the consequences in case the designed and produced systems don't comply with the stated requirements. In particular, the competitiveness in the market of the regional jet aircraft may be negatively affected in case one or more requirements are not verified. The verification of requirement is done through test cases, which belong to different Means of Compliance, as explained in Section III. For example, a test case consisting of an Overall Aircraft Design (OAD) tool can be employed to verify that the mass of the horizontal tail plane is compliant with what stated by the requirement R-0025, as depicted by the "Verification" view of Figure 14.

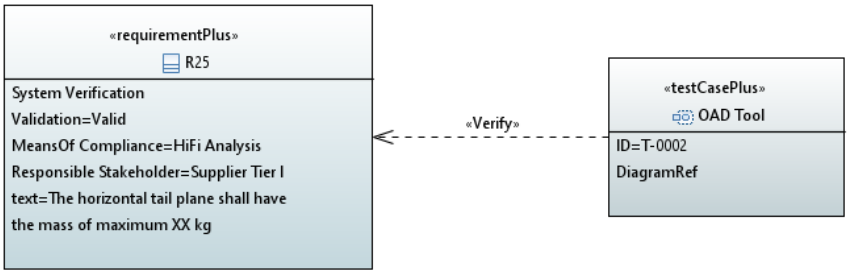

Figure 14 *SysML Requirement Diagram* representing the "Verification" *view* in Papyrus showing which test case can be employed to verify the requirement R-0025.

The last example of *view* automatically obtained through the *MBSE development system* and proposed in the present Section depicts the pattern of the requirement stating the maximum sale price of the regional jet aircraft. The requirement is of the type *design*, and its statement is: "***The aircraft shall have the sale price of maximum XX $ in the market***". The different elements composing this requirement statements can be modeled according to the pattern of design requirements, as represented in the "Requirement pattern" *view* of Figure 15.

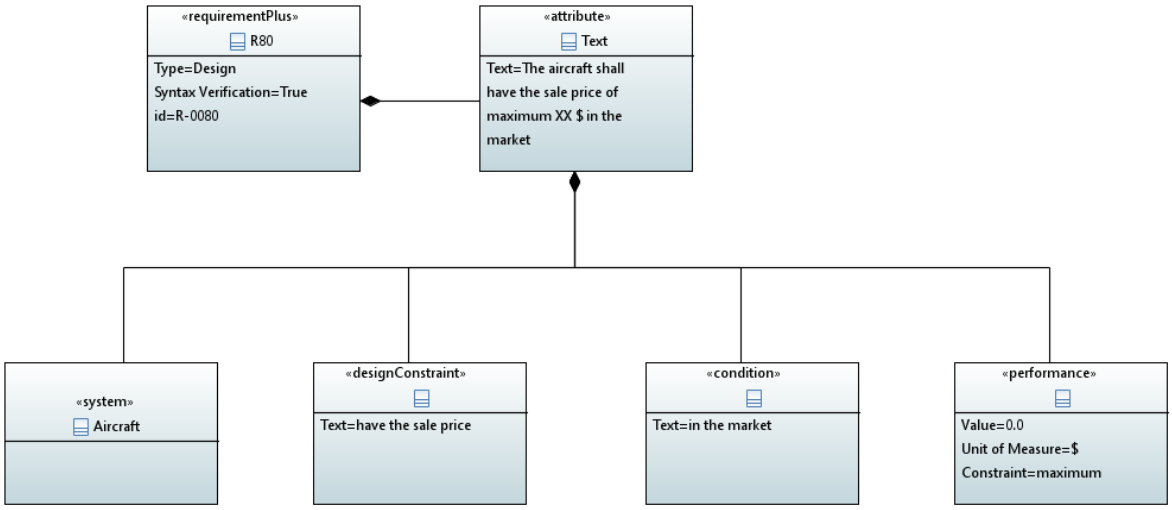

Figure 15 *SysML Requirement Diagram* representing the "Requirement pattern" *view* in Papyrus showing the different elements composing the statement of the requirement R-0080.

A. Advantages and limitations of the model-based architectural framework over a traditional document-based approach

The example of application of the *AGILE 4.0 architectural framework* through its implementation into the *MBSE development system* has demonstrated several advantages of the proposed model-based approach over a traditional document-based one. The most significant advantage is the coherence between the design data and information produced and managed during the development of the *complex systems*. In a model-based approach, all the elements of this data are objects connected together through different kinds of relationship, as *specialization* in the "Stakeholders" *view* of Figure 11, *containment* in the "Needs" *view* of Figure 12, *derive requirement* and *derive consequence* in the "Traceability" *view* of Figure 13, *verification* in the "Verification" *view* of Figure 14 and *composition* in the "Requirement Pattern" *view* of Figure 15. This means that the relation between all the objects is formalized, and it can be easily and quickly assessed how potential design changes are propagated within the model. For example, if a *need* is removed since not valid anymore, it is possible to quickly evaluate which and how *requirements* (and consequently, *design solutions*) are impacted.

A second important advantage is the complete addressing of all the stakeholder needs, which is fundamental for the success of a system, as explained in the Introduction in Section I. The approach of the *AGILE 4.0 architectural framework* aims at formalizing what every stakeholder wants from the system, and how these stakeholder needs are translated into system requirements.

The model of requirement patterns is another advantage of the architectural framework. Since all the words composing the requirement statements can be represented as objects in a model-based approach, they can be re-used in the same model but in different views. For instance, a *function* element of a functional requirement can be included again in the model for the representation of a system functional architecture.

Additional advantages are also present thanks to the *MBSE development system*. For example, scripts for the automatic verification of the model have been developed and integrated, in order to minimize design errors and to accelerate the development process. Moreover, the development system can be accessed by multiple people to whom different tasks are assigned, therefore fostering the collaborative development.

However, the example of application described in this Section has shown some limitations of the proposed approach, but this should encourage additional research activities, within the context of AGILE 4.0 project but also by the entire MBSE community. The main limitation of the proposed model-based approach is represented by the entry barrier that characterizes any novelty. According to the project Consortium, the modeling language (extended SysML) is the main barrier that might hamper the shift from documents to representation through standard models. In order to overcome this limitation, the *MBSE development system* implements tools that automatically create the model from the input specified by the designers. However, knowledge should be acquired by the designers to correctly interpret the *views* generated by the *development system*.

Another significant limitation regards the management of a high quantity of generated data. Although multiple *views* can represent different information of the system being developed, some diagrams might contain a very large quantity of elements, hence negatively affecting the readability of the model. In this regard, it should be considered that a real complex aeronautical product might entail the generation of thousands of requirements, but the MSBE supporting technologies might not be able to clearly represent this large quantity of data. It is opinion of the authors that this limitation has not yet been solved by the MBSE community, and it might negatively hamper the adoption of a model-based approach.

VI. Conclusions and future developments

A new *architectural framework* has been proposed in this paper for the definition and modelling of system stakeholders, needs and requirements through an MBSE approach. This framework represents the main original contribution given by the present paper. The *architectural framework* is being realized within the context of the EU-funded research project H2020 AGILE 4.0 and it exploits MBSE technologies for the realization of an *MBSE development system* and *complex aeronautical systems*. The presented *architectural framework* extends the outcomes of the H2020 AGILE project, where the focus was on the *design and optimization* of complex systems. Further developments of the *architectural framework* are being addressed in the AGILE 4.0 project covering the whole Systems Engineering Product Development process. System functions development, system architecting, value-driven trade off assessment and decision making are other activities of the Product Development process that will be tackled in future by the *architectural framework*.

The proposed architectural framework is being exploited by the AGILE 4.0 project Consortium for the development of seven different Application Cases belonging to the aeronautical domain, but the same modeling guidelines can be adopted by anyone outside the Consortium and doing research and design activities in other

domains, e.g. space, transport and energy. For this reason, the *architectural framework* – e.g. the ontological model for the definition of stakeholders, needs and requirements – has been published as open access. In addition, an extended version of SysML (named *AGILE4Profile*) has been publicly made available through this paper, targeting the modeling of stakeholder, needs and requirements, in compliancy with the prescriptions from INCOSE.

Another contribution of the present paper regards the *MBSE development system*. Its physical implementation derived from the integration of different tools, among which some are owned by project partners, can't be entirely and freely shared outside the Consortium due to Intellectual Property reasons. However, the underlying logical architecture has been described in the paper, and it can be freely instantiated by anyone with different software.

In the final part of the paper, one of the seven AGILE 4.0 application cases has been selected to provide a brief demonstration about the use of the architectural framework and MBSE development system with the modeling of stakeholders, needs and requirements. This example of application has demonstrated many of the advantages about MBSE claimed by the research community. These advantages are fostered by the guidelines and technologies described in the paper, which can be leveraged to increment the *agility* required to streamline, improve and accelerate the definition and modeling of complex systems. However, some limitations have also been identified, which nevertheless can promote additional research activities, performed not only within the project Consortium but also by the entire MBSE community.

Appendix

A new profile derived from SysML and named *AGILE4Profile* has been developed, and it is represented in the *SysML Package Diagram* of Figure 16.

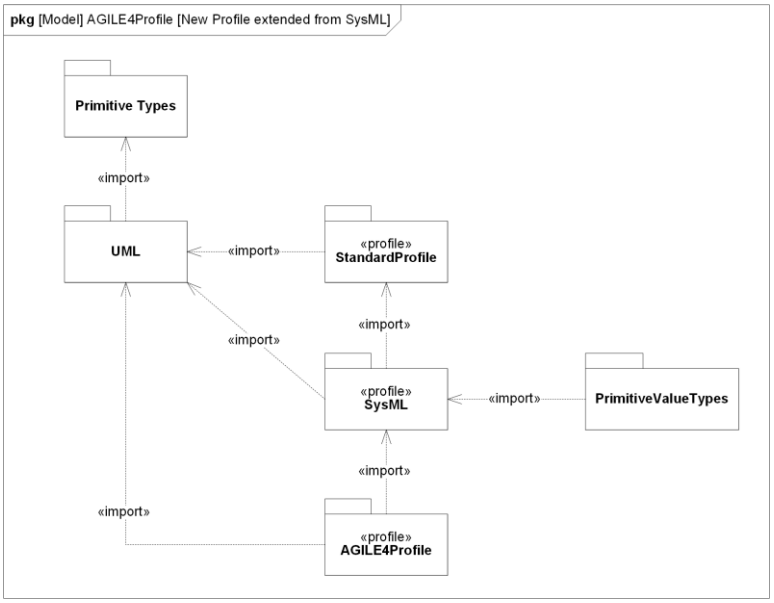

Figure 16 *SysML Package Diagram* representing the extension of the SysML profile (*AGILE4Profile*) adopted in the proposed *architectural framework* for the modeling of stakeholders, needs and requirements.

The new stereotypes introduced with the *AGILE4Profile* are instead collected in the *SysML* diagram of Figure 17. 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 two new stereotypes. The former is ***«need»***, to which the property *Stakeholder* is added. The second stereotype is named ***«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 «*testCase*» is used to generate the new stereotype ***«testCasePlus»***, which adds as information the ID of the specific test case and the name of the SysML diagram realized to described the test case.
- The stereotype ***«deriveCons»*** is generated from the metaclass «*deriveReqt*».
- The metaclass «*block*» is specialized with the new stereotype ***«system»***.

- The following new stereotypes are derived from the metaclass «*block*»: ***«attribute»***, ***«definition»***, ***«consequence»***, ***«function»***, ***«performance»***, ***«timing»***, ***«eventTrigger»***, ***«condtion»***, ***«conditionDuration»***, ***«environment»***, ***«exposureDuration»***, ***«characteristic»*** and ***«designConstraint»***. Each new stereotype adds properties, i.e. text, values, units of measure and constraints (e.g. min, max and equal). Several new stereotypes generated from the metaclass «*block*» are employed to model all the elements belonging to the five patterns addressed in Section III-B.

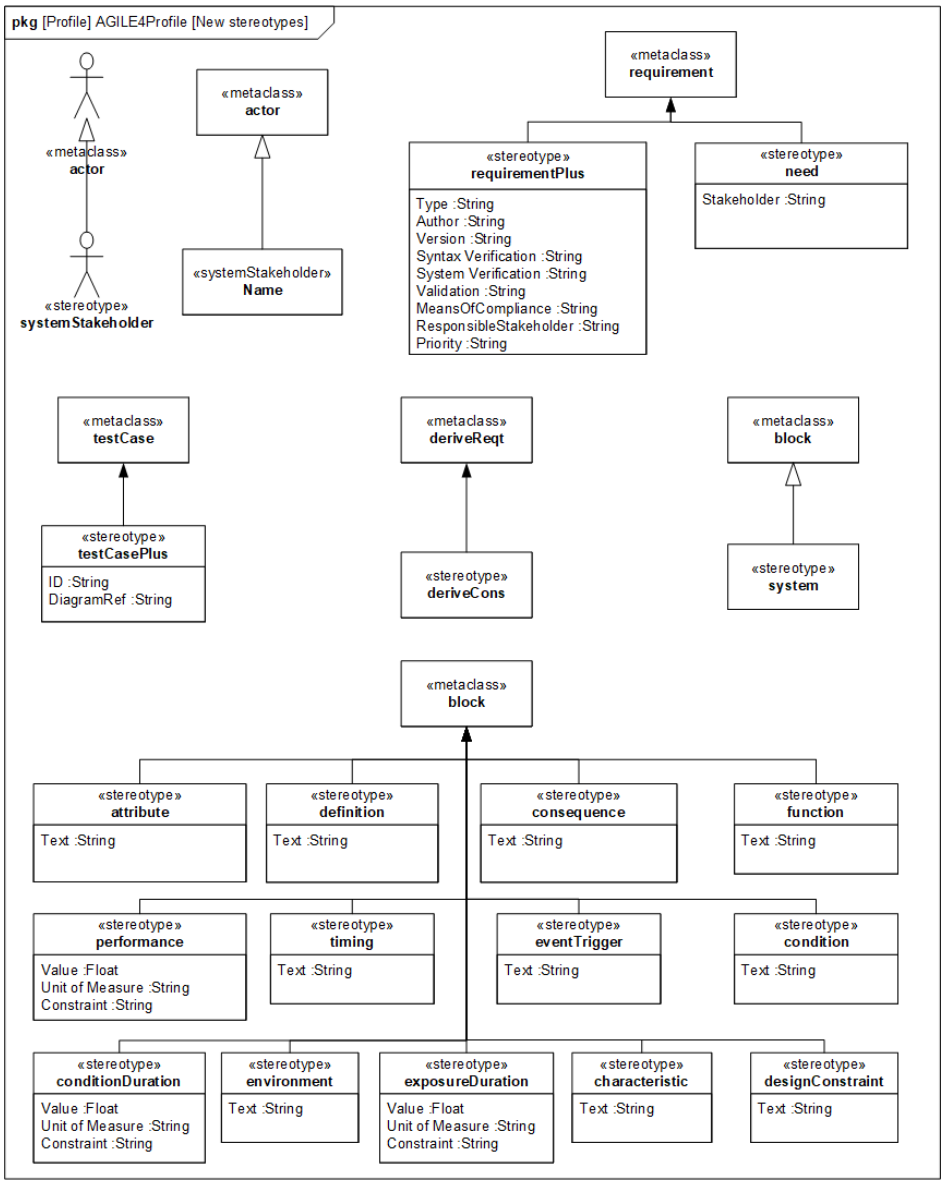

Figure 17 *SysML Package Diagram* representing the new stereotypes for the modeling of stakeholders, needs and requirements introduced with the new *AGILE4Profile*.

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.

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] P. D. Ciampa, G. La Rocca and B. Nagel, "A MBSE Approach to MDAO Systems for the Development of Complex Products," in *AIAA Aviation Forum*, Reno (NV), 2020.

[11] P. D. Ciampa and B. Nagel, "AGILE Paradigm: the next generation collaborative MDO for the development of aeronautical systems," *Progress in Aerospace Sciences*, vol. 119, no. 100643, 2020.

[12] "AGILE Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts," [Online]. Available: http://www.agile-project.eu. [Accessed 2019 March 12].

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

[14] 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.

[15] The Standish Group, "The CHAOS Report," 1994.

[16] INCOSE, "Guide for Writing Requirements, INCOSE-TP-2010-006-01," 2012.

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

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

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

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

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

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

[23] 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.

[24] 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.

[25] G. Génova, J. M. Fuentes, J. Llorens, O. Hurtado and V. Moreno, "A framework to measure and improve the quality of textual requirements," *Requirements engineering*, vol. 18, no. 1, pp. 25-41, 2013.

[26] International Organization for Standardization, "ISO/IEC 29148 FDIS Systems and software engineering - Life cycle processes - Requirements engineering," 2011.

[27] R. Carson, "Implementing structured requirements to improve requirements quality," in *INCOSE International Symposium*, Seattle (WA), 2015.

[28] L. S. Wheatcraft, M. J. Ryan and J. Dick, "On the use of attributes to manage requirements," *Systems Engineering*, vol. 19, no. 5, pp. 448-458, 2016.

[29] M. J. Ryan and L. S. Wheatcraft, "On a cohesive set of requirements engineering terms," *Systems Engineering*, vol. 20, no. 2, pp. 118-130, 2017.

[30] J. Holt, S. Perry, R. Payne, J. Bryans, S. Hallerstede and F. O. Hansen, "A model-based approach for requirements engineering for systems of systems," *IEEE Systems Journal*, vol. 9, no. 1, pp. 252-262, 2014.

[31] US Air Force, "ICAM architecture part II - volume IV, function modelling manual (IDEF0)," 1993.

[32] D. Dori, Model-Based Systems Engineering with OPM and SysML, New York: Springer, 2016.

[33] P. H. Feiler, D. P. Gluch and J. J. Hudak, "The architecture analysis & design language (AADL): An introduction," Carnegie Mellon University, 2006.

[34] Object Management Group (OMG), "OMG UML Profile for DoDAF/MODAF™ (UPDM™)," [Online]. Available: https://www.omg.org/updm/index.htm.

[35] Object Management Group (OMG), "System Modeling Language (SysML)," [Online]. Available: https://www.omg.org/spec/SysML/About-SysML/.

[36] Object Management Group (OMG), "Unified Modeling Language (UML)," [Online]. Available: https://www.omg.org/spec/UML/About-UML/.

[37] Y. Bernard, "Requirements management within a full model-based engineering approach," *Systems Engineering*, vol. 15, no. 2, pp. 119-139, 2012.

[38] A. Salado and P. Wach, "Constructing true model-based requirements in SysML," *Systems*, vol. 7, no. 2, 2019.

[39] B. London and P. Miotto, "Model-based requirement generation," in *IEEE Aerospace Conference*, 2014.

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

[41] *Boggero, Luca, Ciampa, Pier Davide, & Jepsen, Jonas. (2021). AGILE 4.0 MBSE Ontology [Data set]. Zenodo. http://doi.org/10.5281/zenodo.4671896.*

[42] AGILE 4.0 project Consortium, "Zenodo Community Page: AGILE4.0 - Towards cyber-physical Collaborative Aircraft Development," [Online]. Available: https://zenodo.org/communities/agile4. [Accessed 5th May 2021].

[43] G. Donelli, P. D. Ciampa, B. Nagel, G. Lemos, J. Mello, A. Cuco and T. van der Laan, "A Model-Based Approach to Trade-Space Evaluation Coupling Design-Manufacturing-Supply Chain in the Early Stages of Aircraft Development," in *AIAA Aviation Forum 2021*, Washington (US-DC), 2021.

[44] F. Torrigiani, et al., "MBSE Certification-Driven Design of a UAV MALE Configuration in the AGILE 4.0 Design Environment," in *AIAA Aviation Forum 2021*, Washington (US-DC), 2021.