ICAS 2020 SHANGHAI
International Council of the Aeronautical Sciences
32nd Congress of the International Council of the Aeronautical Sciences
September 6-10, 2021
Pudong Shangri-La, Shanghai, China

THE APPLICATION OF THE AGILE 4.0 MBSE ARCHITECTURAL FRAMEWORK FOR THE MODELING OF SYSTEM STAKEHOLDERS, NEEDS AND REQUIREMENTS

Luca Boggero[1], Pier Davide Ciampa[1], Björn Nagel[1]

[1]German Aerospace Center (DLR), Institute of System Architectures in Aeronautics, Hamburg, Germany

Abstract

The present paper shows the application of the model-based *AGILE 4.0 architectural framework* currently being developed within the context of the European Union (EU) - funded AGILE 4.0 research project. This *architectural framework* is conceived to guide designers in the definition, modeling, design and optimization of complex aeronautical systems, by following a Systems Engineering Product Development process. The *AGILE 4.0 architectural framework* supports different activities for the development of systems, such as requirements engineering, functional analysis, system architecting and Multidisciplinary Design and Optimization (MDO). More specifically, the present paper focuses on part of the *architectural framework*, which is being developed to address the initial activities of a Systems Engineering process, namely the definition and modelling of system stakeholders, needs and requirements. The development of this *architectural framework* is indeed fostered by all the potential advantages of Model Based Systems Engineering (MBSE) approaches, which in contrast to traditional document-based approaches might bring to several benefits in the design process, as enhancement of systems design quality, more complete and clearer development of system requirements and specifications and improved communications within the design teams. Seven different application cases are addressed in the paper, demonstrating how the *architectural framework* and its implementation in a *MBSE development system* can be exploited to streamline, accelerate and improve the definition and modeling of complex systems. Advantages but also some limitations of this model-based approach are identified from the applications and addressed in the paper.

Keywords: architectural framework, Model Based Systems Engineering, SysML, requirement, complex system

1. Introduction

The design of a new aircraft is a process that is becoming always more complex, especially due to the development and introduction of new technologies and the continuously increasing demand of higher performance. The aircraft design process is nowadays characterized by a high number of designers and a larger quantity of information and data produced. Design data encompass technical information about the aircraft under design – for instance requirements, specifications, descriptions and interfaces of the several systems, components and parts of the aircraft – but also organizational information produced during the design process, as design decisions, life-cycle of the project, roles and tasks of all the designers involved. Moreover, this huge quantity of data is composed by elements that are connected together through different kinds of relationship. For example, certain system requirements might bring to specific design decisions that entail different solutions. This means, that specific elements of information and data produced and handled in a design project can be impacted in case changes are applied to other elements. Moreover, all the design data are authored by different designers belonging to different organizations of the supply chain or different engineering departments. Therefore, many relationships between the data elements might not be clearly evident, and hence negatively impacting the traceability among all these elements.

With the aim of minimizing the difficulties related to the handling of a large quantity of data produced during the design of an aeronautical product, several organizations and research groups have defined and published *architectural frameworks*. An *architectural framework* is defined by the ISO/IEC/IEEE 42010 standard 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 is a formal description 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 the context of the present paper, 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. Hence, an *architecture* describes the different data elements of the *system.* A *system architecture* can depict for example all the parts that compose the *system*, its life-cycle, its operations and many other details. An *architectural framework* provides the guidelines for the standard representation of the multiple *system architectures.*

As mentioned before, several *architectural frameworks* have been proposed and made available in literature, e.g. Zachman's Framework [3], Department of Defense Architectural Framework (DoDAF) [4], British Ministry of Defence Architecture Framework (MODAF) [5], NATO Architectural Framework (NAF) [6] and The Open Group Architecture Framework (TOGAF) [7]. Among all of them, the *architectural framework* being developed in EU funded research project H2020 AGILE 4.0 [8] is presented in this paper. The AGILE 4.0 project, led by DLR, expands the ambitions of its predecessor H2020 AGILE [9] project, and it targets new methods and technologies to improve, streamline and accelerate the development of complex systems, addressing all the main pillars of the aeronautical supply-chain: design, production, certification and manufacturing. The *AGILE 4.0 architectural framework* is supported by novel Model Based Systems Engineering (MBSE) technologies. Indeed, traditional aircraft development activities performed along the different stages of a Systems Engineering Product Development process (e.g. identification of stakeholder needs, determination of system functions, collection of requirements, verification and validation tasks) are addressed in numerous documents, which collect the whole quantity of data and information produced. This traditional document-based approach is negatively impacted by many limitations and disadvantages, including for example poor traceability, lack of clarity, misunderstandings, and ambiguity. A new approach based on models instead would improve all the design activities, enhancing the quality of the process and results, and facilitating the clear and correct communications within the design team [10]. Due to all these potential advantages, several companies, many of them operating in the aerospace domain, have already started the transitioning towards innovative models-based approaches, and therefore MBSE is expected to play an increasing role in the field of Systems Engineering in the next decades [11]. All these motivations are pushing the AGILE 4.0 project Consortium to the development of an *MBSE architectural framework*, which would provide clear guidelines for the representation of multiple *system architectures* through models.

The AGILE 4.0 project is still running, since it was started in September 2019 and it will last for three years. Therefore, the *AGILE 4.0 architectural framework* is still under development, but it can be currently employed for the definition and modeling of stakeholders, needs and system requirements. Moreover, the project Consortium is implementing the *architectural framework* into a *development system*, which includes several technologies that can be used to accelerate and improve the design and operation of complex aeronautical products. In addition, it is worth noting that a *development system* itself can be designed and built by leveraging MBSE approaches, as explained by Ciampa *et al* [12]. Therefore, also the *AGILE 4.0 development system* is being realized by exploiting model-based approaches and technologies [13]. The present paper aims at proposing some examples of application of the *AGILE 4.0 architectural framework* employed through the *AGILE 4.0 MSBE development system*. More specifically, this paper shows how the different application cases addressed in the project are tackled by employing the novel MBSE technologies being developed in the project. Ultimately, the present paper collects the main feedback provided by several partners about the novel model-based approach, but also some limitations.

In order to reach the previously stated objectives, this paper is organized as follows. After the introduction of Section 1, an overview of the *AGILE 4.0 architectural framework* and its implementation into the *MBSE development system* are presented in Section 2. The seven application cases tackled in the project and addressed in the paper are introduced in Section 3. Section 4 provides a brief example of definition of stakeholders, needs and requirements, supported by documents, highlight the limitations and disadvantages of this legacy approach. The most important part of the present paper is Section 5, where the *AGILE 4.0 architectural framework* and *MSBE development system* are employed to model the seven application cases. This paper ends with Section 6, in which the main feedback gathered from the different project partners is reported.

2. AGILE 4.0 architectural framework and MBSE development system

The present section aims at providing a short introduction about the *architectural framework* and

MBSE development system being addressed in the AGILE 4.0 project. More details are written in [14].

As mentioned in Section 1, the current version of the *AGILE 4.0 architectural framework* and its implementation into the *MBSE development system* guide the definition and modeling of stakeholders, needs and requirements. A **stakeholder** is 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" [15]. Given an aircraft as a system, examples of stakeholders encompass: Original Equipment Manufacturers (OEMs), suppliers, regulation authorities, airlines, flight and ground crews, passengers, maintainers, and air traffic controllers. All the stakeholders express different wishes, necessities and desires from the system. In other words, stakeholders have different **needs**, which are defined as "informal expressions of something that has to be provided, ensured or avoided by a system or the development project of this system" [16]. For instance, airlines want to maximize profit, and passenger would instead demand a comfortable flight. Needs are generally unstructured and expressed in fuzzy or general, ambiguous terms, and therefore they must be translated into **requirements**, which instead should follow precise patterns and rules to assure characteristics as unambiguity, completeness, feasibility, verifiability and correctness. A requirement is "a statement which translates or expresses a need and its associated constraints and conditions" [16].

As any *architectural framework* [17], the one being developed in AGILE 4.0 is composed by *ontologies* and *viewpoints*. An **ontology** defines all the main concepts of the system architecture and the relationships between them. A **viewpoint** instead lists all the conventions for the construction, interpretation and use of system architectures from the point of view 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.

Figure 1 shows through a *SysML (Systems Modeling Language) Internal Block Diagram* the ontology for the definition of the main concepts previously introduced and their relationships: complex system, MBSE development system, stakeholders, needs, requirements. Moreover, the ontology includes other concepts that are explained below, as requirement types, rules and attributes. This ontology developed within the context of the AGILE 4.0 project is published as open access [18] on the Community page of the project on the Zenodo website [19], from where it can be freely downloaded and re-used by any user inside and outside the project Consortium. The available files represent the meta-models, rendered by OWL (Web Ontology Language), supporting the development of any complex system in any domain.

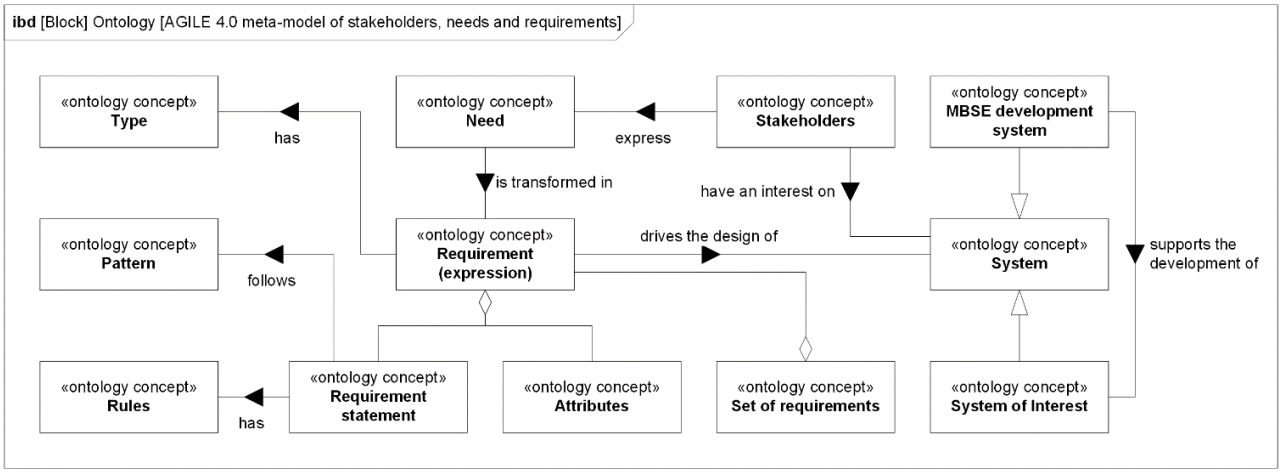

Figure 1 – *SysML Internal Block Diagram* representing the ontology of the *AGILE 4.0 architectural framework* representing the main concepts and their relationships for the definition of system stakeholders, needs and requirements [14].

One of the key concepts of the ontology is the *system*. Within the specific context of the present paper, the system is a complex aeronautical product (e.g. an aircraft), but according to its generic definition, also the *MBSE development system* is a system, which aims at supporting the designer in the development of the *system of interest* (e.g. through tools and file formats). The development of the system begins with the identification of the *stakeholders*, which have an interest on the system during its life-cycle, and therefore which express various *needs*. These *needs* have to be transformed into *requirement (expressions)*, which drive the design of the system. Requirements are composed by two main parts: *requirement statement*, i.e. the text, and a series of *attributes* required for the optimal management of requirement. *Attributes* are recommended and explained in [20], and examples

encompass: author of requirement, identification number, and test cases. Moreover, multiple requirements can be grouped within same *sets of requirements*, for instance when dealing with the same subject. Differently from needs, which are generally incomplete, unclear and ambiguous, requirement statements have to follow predefined *rules* that assure quality characteristics as unambiguity, completeness, verifiability. A long list of rules is prescribed by the International Council On Systems Engineering (INCOSE) and published in [20]. Finally, requirement statements have to follow different *patterns*, i.e. they have to contain mandatory and optional elements including for instance functions, performance characteristics, durations and conditions. The patterns depend of the type of the requirements, e.g. functional or performance requirements.

Other than ontologies, an *architectural framework* is composed by viewpoints. Ten different viewpoints are part of the *AGILE 4.0 architectural framework* 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* [21] depicted in Figure 2 collects all the ten viewpoints.

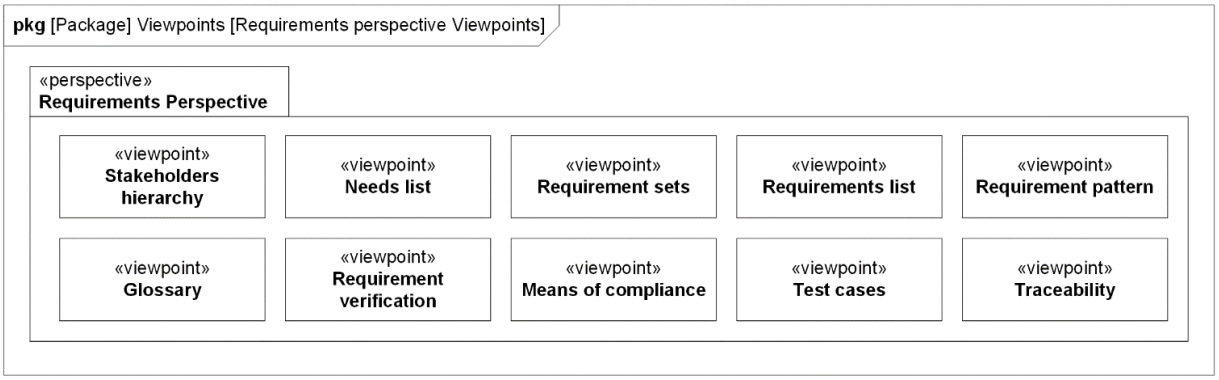

Figure 2 – Viewpoints of the *AGILE 4.0 architectural framework* belonging to the requirement perspective. A *SysML Package Diagram* is adopted to model the viewpoints [14].

The ten viewpoints of the *AGILE 4.0 architectural framework* provide the following information:

- **Stakeholders hierarchy**: representation of all the stakeholders that have an interest with the system under design. The hierarchy among stakeholders is shown as well by views compliant with this viewpoint.
- **Needs list**: representation of the needs expressed by the different stakeholders. Views compliant with this viewpoint should show which stakeholders are expressing which needs.
- **Requirement sets**: list of all the considered sets of requirements. Views compliant with this viewpoint don't show any other detail about the requirements belonging to the different sets.
- **Requirement lists**: collection of requirements, for each of them reporting information as requirement statement, identification number, requirement type, author and version.
- **Requirement pattern**: representation of all the pattern elements (e.g. function, performance, condition) of each requirement statement.
- **Glossary**: explanation of specific nomenclature that is represented in all the views.
- **Requirement verification**: representation of means of compliance and test cases used to verify a specific requirement, i.e. to prove that the system is compliant with what expressed by the requirement.
- **Means of compliance**: list of all the generic means (e.g. "analysis", "simulation") employed for the verification of requirements.
- **Test case**s: description of a test case (i.e. a specific means to verify a requirement, for example a specific software) used to prove the compliancy with a requirement.
- **Traceability**: representation of all the "derivation" relationships between needs and requirement and between multiple requirements.

More details about the ten viewpoints of the *AGILE 4.0 architectural framework* are reported in [14]. The present paper follows the viewpoints to realize architectural views of the seven AGILE 4.0 application cases. The obtained views are addressed in Section 5. All the views have been automatically derived through the *MBSE development system*, which implements all the viewpoint just explained. The *MBSE development system* integrates different tools available in literature or developed the project Consortium. Figure 3 shows the physical architecture of the *MBSE*

development system, consisting of the integrated tools for the definition and modeling of stakeholders, needs and requirements. The platform KE-chain[1] provided by the company KE-works serves as front end of the *MBSE development system* and as a process modeler. It is accessed by multiple users to define data of the system under development, as stakeholders and needs. In addition, it is used for the elicitation and development of requirements (including the definition of all the requirement attributes), and connection between needs, derived requirements and originating stakeholders. The collected data can also be inspected in KE-chain by other users for validation purposes. Moreover, automation modules are integrated with KE-chain to verify the completeness and consistency of the model. Once the model is created, it can be exported and visualized in the form of SysML diagrams, which are automatically built by the tool MBSElib, developed by DLR, according to the viewpoints prescribed by the *AGILE 4.0 architectural framework*. The views generated with MBSElib are saved in a Papyrus[2] project file and can be opened in Papyrus for their inspection and validation.

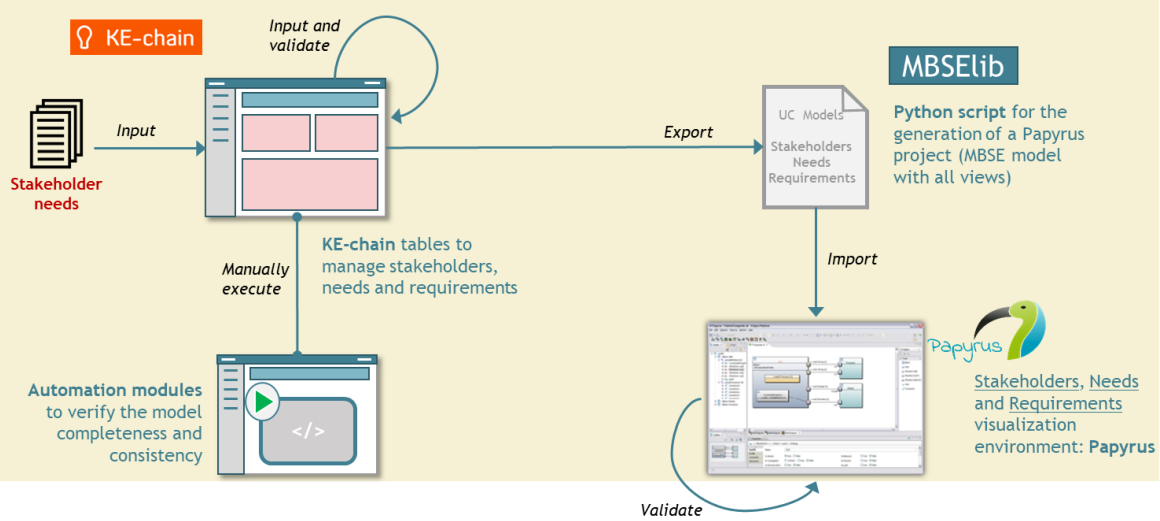

Figure 3 – Architecture of the MBSE development system for the definition and modeling of stakeholders, needs and requirements [14].

3. Introduction to the AGILE 4.0 Application Cases (AC)

Seven different application cases are designed in the AGILE 4.0 project through the *AGILE 4.0 architectural framework* and the *MBSE development system*. The application cases are validated by industrial partners of the project Consortium, and they focus on different scenarios, covering the whole development life-cycle, including *production*, *certification* and *upgrade* (see Figure 4). A brief description of the seven application cases is provided in this section.

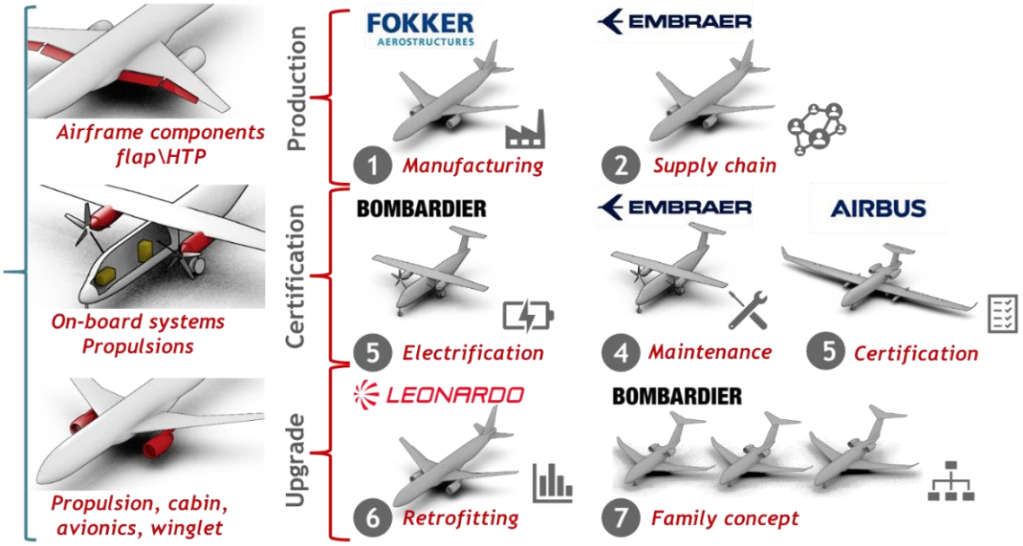

Figure 4 – Seven application cases described in the paper addressing different scenarios of the aircraft development life-cycle.

[1] https://ke-chain.com/?lang=en
[2] https://www.eclipse.org/papyrus/download.html

3.1 Application Cases with focus on Production

The production phase of the development life-cycle is the focus of two application cases, referred respectively as AC1 and AC2. A 90-passenger regional aircraft is taken as reference for both the cases, but two different aircraft systems are tackled. AC1 specifically addresses the optimization of trailing edge flaps while taking manufacturing aspects into account. AC2 instead models and analyzes the production of Horizontal Tail Planes by different industrial supply chains, made of an Original Equipment Manufacturer but different combinations of suppliers at different tier levels. More information about AC2 can be found in [22].

3.2 Application Cases with focus on Certification

Three application cases (AC3, AC4 and AC5) focus on certification. The main objective of these application cases is to include certification aspects during the design process. Novel and innovative architectures of on-board systems (including propulsion systems) are investigated by the three application cases. In particular, AC3 aims at identifying and evaluating multiple system architectures driven by safety constraints while AC4 is focused on the continuous airworthiness, which mainly defines the maintenance process and aircraft maintainability. Both the application cases are based on a 19-passenger regional aircraft. AC5 instead addresses part of the certification process of systems and airframe (e.g. electromagnetic compatibility) of Unmanned Aerial Vehicles (UAVs). More information about AC5 can be found in [23].

3.3 Application Cases with focus on Upgrade

Upgrade aspects are finally addressed by the last two application cases. The case referred to as AC6 deals with the “retrofitting” of an existing 90-passenger regional jet aircraft, by re-designing and re-integrating novel and improved versions of engines and on-board system architectures. AC7 instead tackles the development of a business-jet family compose by three 8-passenger aircraft with different cabin length and design range.

4. Document-based approach: application and limitations

A document-based approach was initially adopted by the application cases to identify stakeholders, collect their needs, and develop system requirements. Processes and guidelines required to accomplish these tasks were presented to the partners involved with the application cases. In particular, explanations on how to write requirements (see pattern, attributes and rules in Section 2) were provided, but the partners were asked to collect the information and results in documents. Figure 5 shows an example of requirements collection supported by a document-based approach in AC2. The document collects some requirements focusing on the aircraft level. The requirement statements are manually written adhering to the patterns. Different colors are employed to highlight the different text elements of the statements, for instance red to highlight functions and brown to highlight performance characteristics. Other than the requirement statements, all the attributes are reported in different columns, including other requirements or needs originating each requirement (column D).

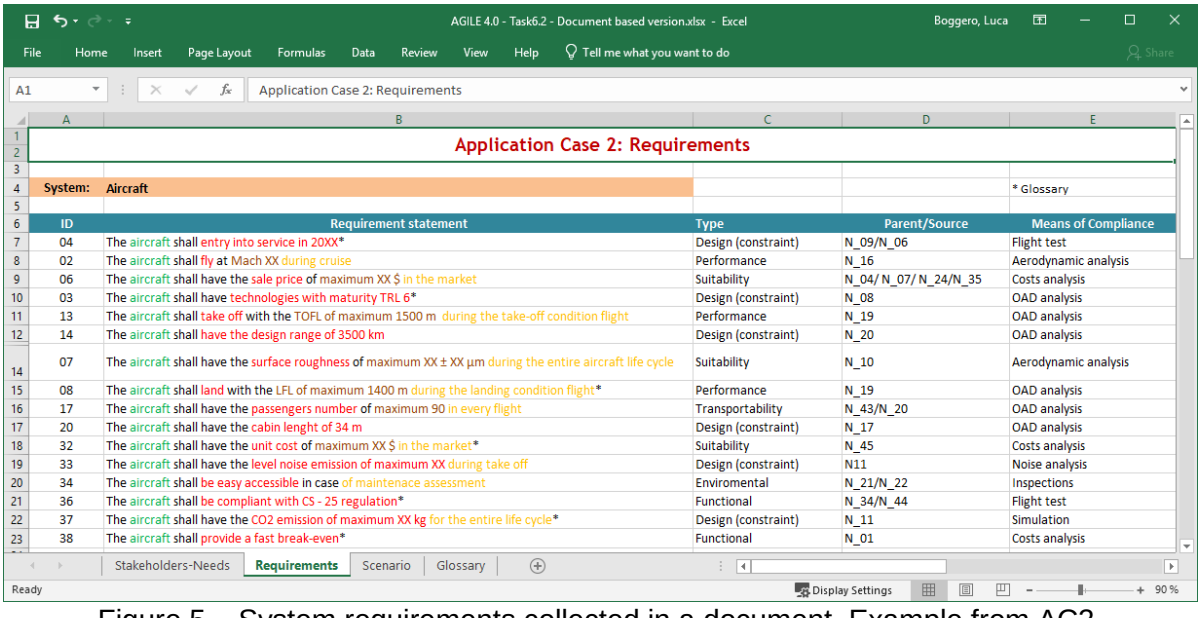

Figure 5 – System requirements collected in a document. Example from AC2.

The following disadvantages and limitations of this document-based approach were found by the different application case partners:

- The activity of reporting information in documents is prone to errors. Only manual verification of the collected information can be done, but this can take a lot of time and effort and it can be ineffective.
- Traceability between all the collected data and information is negatively impacted by the document-based approach. This implies difficulties when applying modifications, since it would be challenging to clearly assess what and how is impacted by design changes.
- All the elements inside the requirement statements are “pieces of text”, and not “objects” of a model. Therefore, in a document-based approach these elements can’t be are re-used in other phases of the development. For instance, performance characteristics expressed in performance requirements can also be results of MDO processes. If these elements were “objects” instead of “text”, they could be employed to verify the requirements.
- It is difficult to adhere to patterns when writing requirement statements in documents. This would entail errors that would negatively affect the quality of the requirement, and therefore jeopardize the success of the project.
- The joint effort between multiple people collaborating in the same project would be undermined by the document-based approach.
- It is difficult to re-use in a new and different project all the results collected in documents. In case part of the data and information generated within the context of a project was exploited in a new study case, a drastic reduction of development effort and time would be achieved.

All these limitations and disadvantages have motivated the project Consortium to adopt a new approach based on models instead of documents. This new model-based approach is guided by the *AGILE 4.0 architectural framework* and supported by the *MBSE development system*. The most significant results determined during the development of the seven application cases through a model-based approach are described in the following section.

5. Model-based approach: application

Other than in documents, the entire design information that is generated and handled in projects can be more effectively collected in models, thus overcoming all the limitations and disadvantages previously presented concerning a document-based approach. The present section shows some examples of models addressing the stakeholders, needs and requirements of the seven AGILE 4.0 application cases.

5.1 Stakeholders and Needs model

The first model proposed in this paper represents system stakeholders and their needs. As reference example, AC1 is chosen. Since this application case focuses on the design and production of flaps, three main types of stakeholders are identified, as represented in the *SysML Block Definition Diagram* of Figure 6 that has been automatically generated through the *MBSE development system*. Three representatives of a supply chain are considered, each one belonging to a different tier level. The top level of the supply chain is occupied by the OEM, which targets the design production and assembly of the overall aircraft, except the flaps. The flaps are indeed designed and produced by a Flap Manufacturer, which belongs to the tier-1 level of the supply chain. In turn, the Flap Manufacturer outsources to realization of smaller components (e.g. hinges, tracks) to multiple lower level suppliers. Additional stakeholders are considered in AC1, as Certification Authorities, the Government and Aircraft Users.

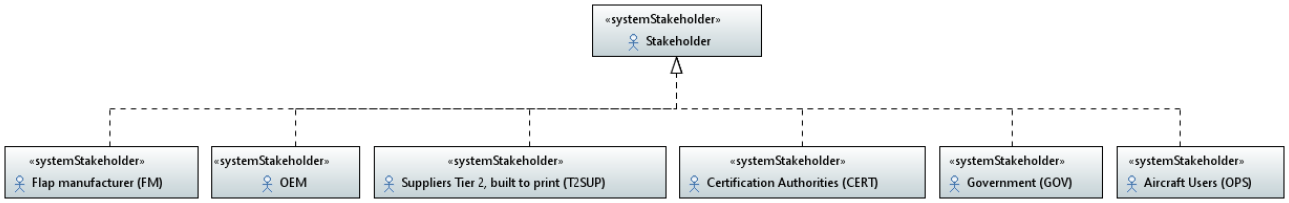

Figure 6 – System stakeholders model represented by a *SysML Block Definition Diagram*. Example from AC1 [24].

All the stakeholders express different needs. Some examples of needs collected from the OEM are

modeled and represented in the *SysML Requirement Diagram* of Figure 7. These needs are specifically addressing the flap, and they prescribe for instance: flap geometry and dimensions (N-0011, N-0031), delivery time (N-0012), development costs (N-0013), mass and structural characteristics (N-0014, N-0016).

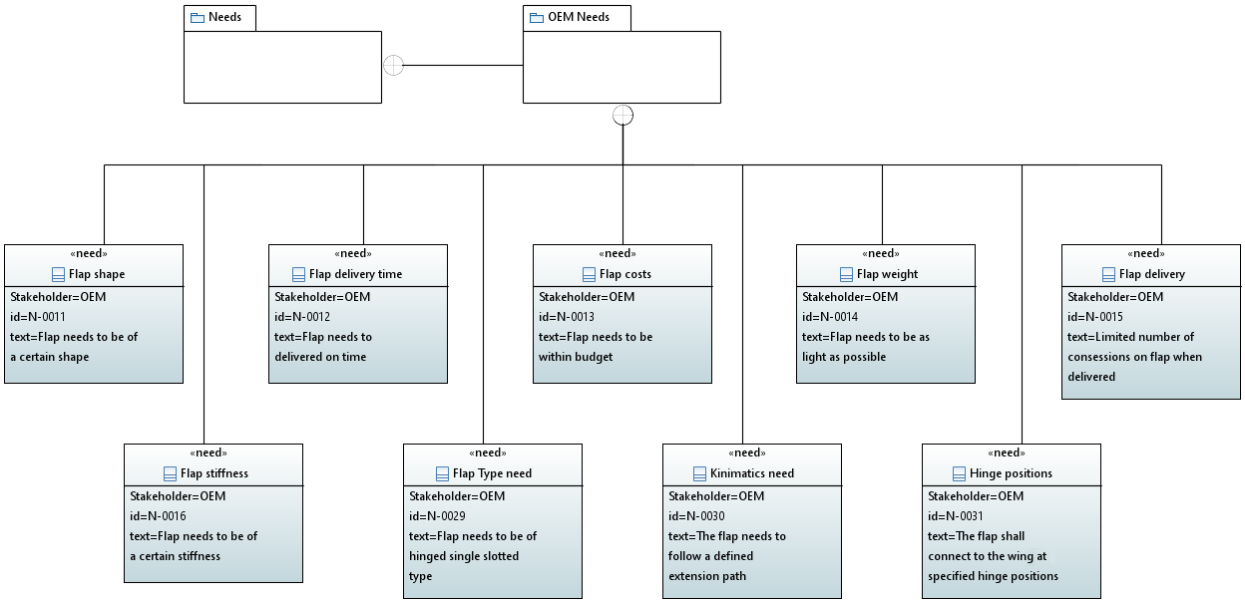

Figure 7 – Stakeholder needs model represented by a *SysML Requirement Diagram*. Example from AC1 [24].

5.2 Operational scenarios model

After the collection of stakeholder needs, some of them can be validated through operational scenarios. “Validation” means to assure that the system resulting from the design will reflect the stakeholder needs. It is therefore important for the designers to correctly understand the expectations of all the stakeholders. An operational scenario is indeed a means of communication between the designers – who are designing the system – and the stakeholders, which are going to operate the system. The example of operational scenario model proposed below regards AC7, which focuses on the development of a fleet of business jets. The *SysML Sequence Diagram* depicting one of the many operational scenarios of this system is reported in Figure 8. The “system” represented in this scenario is the fleet of the airplanes composing the aircraft family.

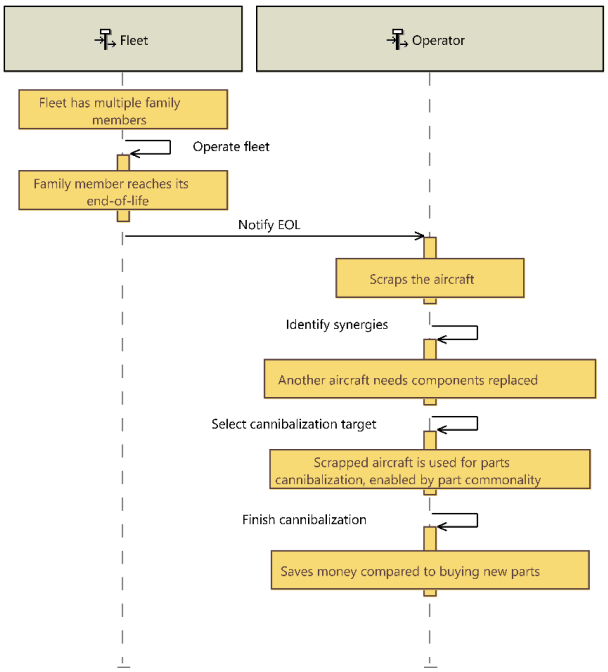

Figure 8 – Operational scenario model represented by a *SysML Sequence Diagram*. Example from AC7 [25].

The model of this scenario describes an operation of aircraft “cannibalization”. After an aircraft of the family reaches its end-of-life, part of it might still be good and may be used to replace parts on another aircraft of the same family (so not necessarily of the same exact type). This operation satisfies the need identified by the operator about the reduction of operating costs. Since components that are still working can be taken from another aircraft of the same family, there’s not the necessity of purchasing new items, and therefore maintenance costs can be limited.

5.3 Requirements model: list

Once needs have been collected and validated through operational scenarios, they should be transformed into system requirements. Figure 9 shows through an automatically generated *SysML Requirement Diagram* the model with aircraft requirements developed for the AC5 and dealing with certification aspects. It can be noted that the model can be created and handled by multiple people, and therefore different requirements might be generated by different people. In this specific example, aircraft certification requirements are defined by two partners of the AGILE 4.0 project Consortium: the DLR and the industrial partner Leonardo (LDO).

As explained before, the present view shows only part of the total information regarding requirements. Additional information is represented in other views, as explained in the following subsections.

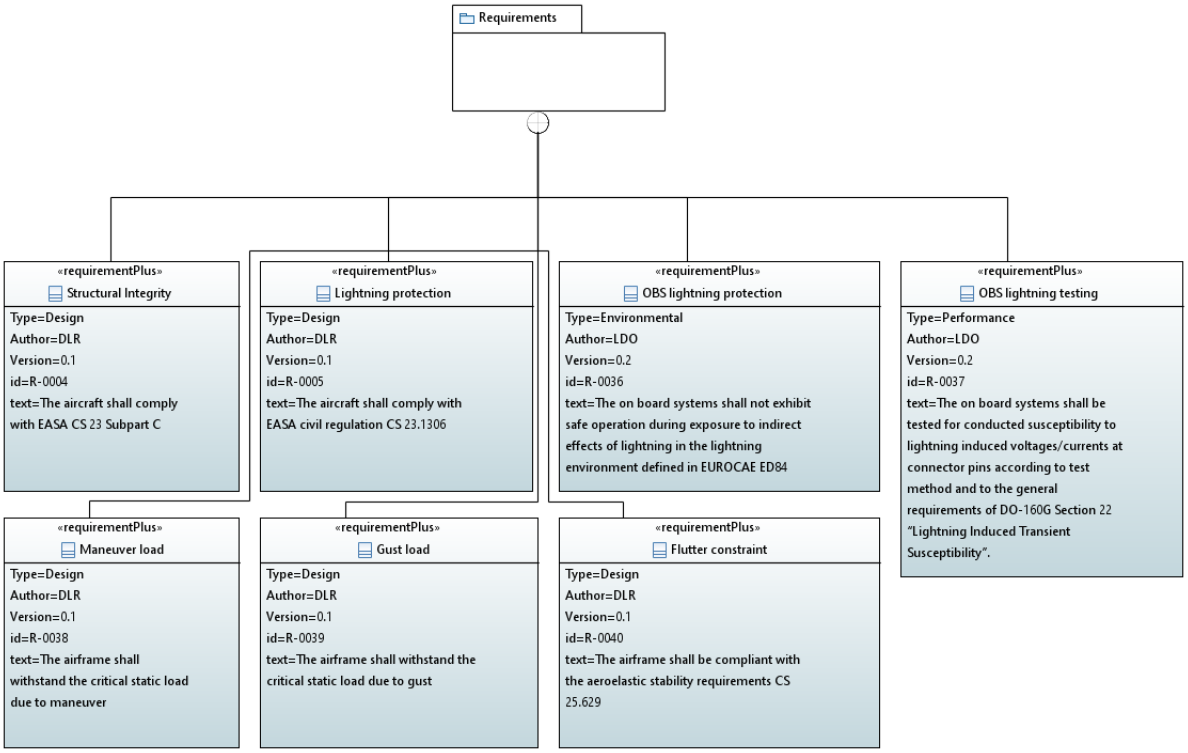

Figure 9 – Requirements list model represented by a *SysML Requirement Diagram*. Example from AC5 [26].

5.4 Requirements model: patterns

The transition from a document to model-based approach tackled through the *AGILE 4.0 architectural framework* regards also the modeling of requirement statements. The solution adopted in the project is to represent through objects all the main elements (i.e. words) that compose the text of a requirement. In addition, all these elements together form the patter of the requirement, which is a pre-defined structure that requirements have to comply with, in order to have good requirement qualities, such as completeness, correctness, unambiguity and verifiability. The *SysML Requirement Diagrams* in Figure 10 show the model of the requirement statement of two different requirements, instantiating the elements that characterize the two different patterns. More specifically, Figure 10 (a) represents a functional requirement, whose pattern is composed by two elements: the system – i.e. the aircraft – and the function, which is “fly safely”. Therefore, the first requirement statement is “the aircraft shall fly safely”. Analogously, the second requirement represented in Figure 10 (b) is composed by all the elements characteristic of “Design” requirements, i.e. system (“the aircraft”), design constraint (“have the sale price”), performance (“of maximum TBD $”) and condition (“in the market”).

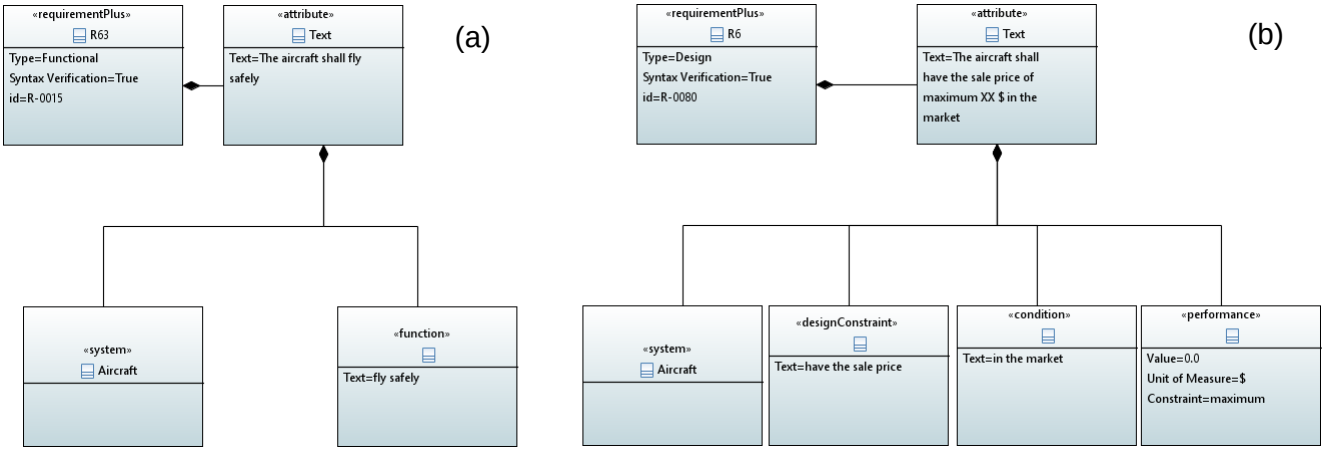

Figure 10 – Model of (a) functional and (b) design requirement patterns represented by *SysML Requirement Diagrams*. Examples from AC2 [24].

5.5 Requirements model: verification

An important activity of any product development process consists of the verification of requirements, i.e. to check and prove that the product being designed is compliant with the stated requirements. Therefore, the requirements model prescribed by the *AGILE 4.0 architectural framework* should also be generated to indicate how requirements will be verified. Figure 11 shows the "Requirement verification" view (depicted through *SysML Package Diagrams* that have been automatically generated by the *MBSE development system*) of two different requirements. The first requirement (a) is selected from AC3, and it deals with the safety characteristics of an Environmental Control System. In this case, a specific certification (safety) tool is identified as a test case, which belongs to the Means of Compliance "high-fidelity analysis". The second requirement instead is from AC4, addressing the dispatch reliability of the aircraft under design. A maintenance tool that also belongs to the Means of Compliance "high-fidelity analysis" is used to verify this requirement.

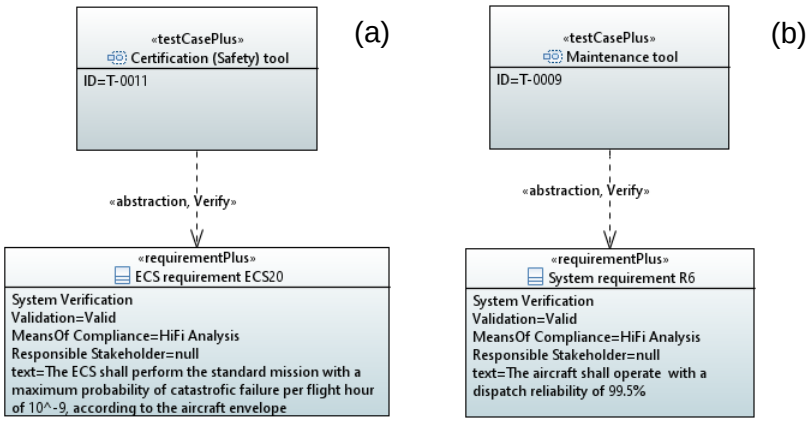

Figure 11 – Requirement verification Model (*SysML Package Diagrams*) showing the test cases for the verification of requirements of AC3 (a) and AC4 (b) [26].

5.6 Traceability model

The last view presented in this paper is named "Traceability", and it shows through *SysML Requirement Diagrams* which requirements are derived from which needs or other requirements, and which are the consequences that might happen in case a requirement is not verified. The traceability model is very important to evaluate how needs, requirements and consequences are related together. Therefore, it is also important to have evident which are the sources of the different requirements, and which impacts these requirements have onto the project in case they are not fulfilled. The example proposed in this paper is based on AC6, and a portion of its relative traceability view is depicted in Figure 12. As all the views described in the present paper, also the "Traceability" view has been automatically generated through the *MBSE development system*. A key requirement of this application case states that the aircraft should generate low emissions in any flight condition. This requirement is derived from more than one need dealing with "green" and clean aviation specified by different stakeholders. This requirement generates other requirements at aircraft-level, specifically

prescribing the minimum reductions of polluting emissions (both CO_2 and NO_X) and noise. In case these requirements weren't fulfilled, the aircraft wouldn't be compliant with regulations and wouldn't be attractive for passengers and airlines.

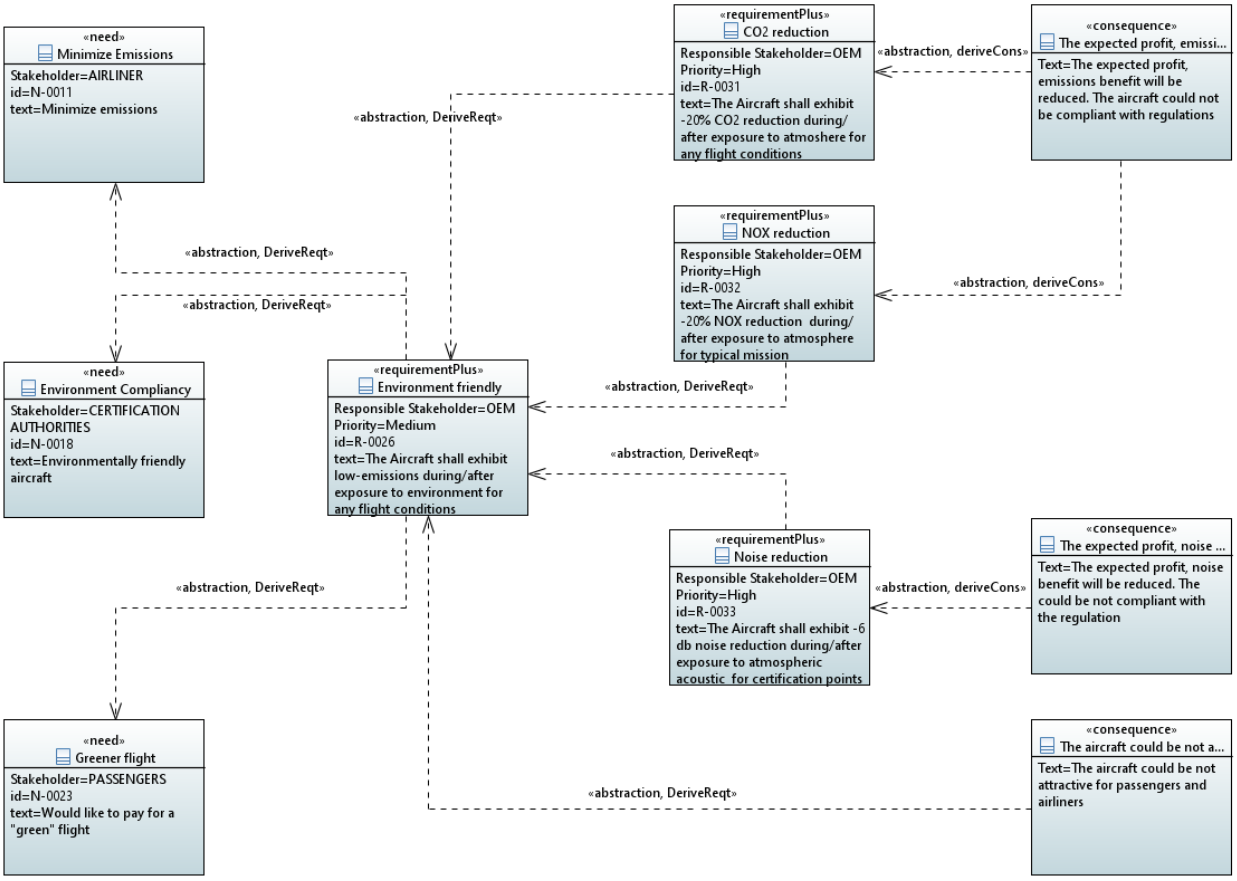

Figure 12 – Traceability model representing "derivation" relationships between needs, requirements and consequences, represented by a *SysML Requirement Diagram*. Example from AC6 [25].

6. Conclusions: advantages and potential improvements of a model-based approach

The present paper focused on the activities performed within the context of the EU-funded H2020 AGILE 4.0 research project dealing with the transition from a traditional document-based approach to an innovative model-based approach. More specifically, the paper has shown some examples of application of the *AGILE 4.0 architectural framework,* and its implementation in the *AGILE 4.0 MSBE development system*. The current versions of *architectural framework* and *development system* are employed to identify and model system stakeholders, their needs and system requirements.

Several advantages have been identified from the application of the model-based *AGILE 4.0 architectural framework*. The proposed framework and the *development system* indeed exploit modelling methods and technologies to improve the *agility* required during the definition phases of complex systems. In other words, these solutions foster the collaboration among multiple designers and integrate various automatisms to streamline, improve and accelerate the development of complex systems.

The most relevant advantage regards the coherence between all the data and information developed and handled during the design process, since all this data in a model-based approach is represented by objects connected together (e.g. through "derivation" relationships in the "traceability" view in Figure 12). In other words, the traceability is improved thanks to a model-based approach, and this can be helpful in case design changes are made and should be propagated to the rest of the model.

Another important advantage is the complete addressing of all the stakeholder needs, which is one of the key activities for the success of a product. Thanks to the approach of the *AGILE 4.0 architectural framework*, it is evident what every stakeholder wants from the system, and how these stakeholder needs are translated into system requirements. This information is represented in the "stakeholder" and "need" models (Section 5.1) and in the "traceability" model (Section 5.6). Moreover, stakeholder needs can be easily validated through model-based scenarios, which can be more comprehensible than descriptions reported in documents.

The *AGILE 4.0 architectural framework* prescribes also guidelines about the patterns of the different requirement statement. This is another significant advantage of a model-based approach, because all the words composing the text of requirements are represented as objects, which can be re-used in the same model but in different views. For example, the same “function” element of a functional requirement can be re-used again in the model within the representation of a system functional architecture.

Other important advantages of the proposed model-based approach are fostered by the *MSBE development system*. First, scripts are part of the *development system* to automatically verify the correctness of the entire information. This can be possible since the information is represented by models rather than being collected into documents. Second, the *MBSE development system* can be accessed by multiple people with different roles. Therefore, the system model can be collaboratively generated and handled.

Nevertheless, some limitations of the model-based approach have been encountered, which can however promote further initiatives of further developments and improvements. The main disadvantage of this approach is represented by the entry barrier that characterizes any novelty. This entry barrier is reflected by the modeling language and by the new tools that the designers have to use. The *MBSE development system* aims at overcoming this limitation by implementing tools that automatically create the model (represented in SysML) from the input specified by the users. However, knowledge should be acquired by the users to correctly interpret the views generated by the *development system*. The second main limitation regards the difficulty of managing a high quantity of generated data. Although the different views can represent part of the information of the system under development, some resulting diagrams might contain a large quantity of elements, hence negatively affecting the readability of the model. For instance, thousands of requirements can be generated during the development of a real complex aeronautical product, and the MBSE supporting technologies should be able to clearly represent this large quantity of data. However, this limitation seems still unsolved within the entire MBSE community, and it might negatively hamper the adoption of a model-based approach.

Finally, it is worth mentioning here some future activities that will extend the scope of the *AGILE 4.0 architectural framework*. The technologies introduced and applied in this paper focus on the definition of system stakeholders, needs and requirements, but additional guidelines are under development within the context of the AGILE 4.0 project to address system architecting activities and for the verification of requirements through MDO processes.

7. Acknowledgements

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 the partners of the AGILE 4.0 consortium for their contribution and feedback.

8. Contact Author Email Address

Luca Boggero

Institute of System Architectures in Aeronautics, German Aerospace Center (DLR)

Hein-Saß-Weg 22, Hamburg (Germany)

Luca.Boggero@dlr.de

9. Copyright Statement

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

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

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

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

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

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

[19] 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].

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

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

[22] 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* , Washington (US-DC), 2021.

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

[24] AGILE 4.0, "D6.3 Production driven stream use cases – model based definition," AGILE 4.0 project (H2020-815122), 2021.

[25] AGILE 4.0, "D8.3 Upgrade driven stream use cases – model based definition," AGILE 4.0 project (H2020-815122), 2021.

[26] AGILE 4.0, "D7.3 Certification driven stream use cases – model based definition," AGILE 4.0 project (H2020-815122), 2021.