PROCESSES, METHODS AND TOOLS SUPPORTING THE DEVELOPMENT OF AERONAUTICAL SYSTEMS

Luca Boggero[1], Jasper Bussemaker[1], Giuseppa Donelli[1], Francesco Torrigiani[1] & Björn Nagel[1]

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

Abstract

Various processes, methods and tools are needed to support the development of new aeronautical systems. Those should cover the whole life-cycle, from the definition of stakeholders to system design and optimization, manufacturing and, eventually, end of life. Since years, the Digital Development Process group at the German Aerospace Center (DLR) is doing research on processes, methods and tools supporting and enhancing the development of new systems. Some of them are taken from the literature, others are developed by the research group. This paper presents an overview of these processes, methods and tools, highlighting how they can support and improve the main activities of a system development process.

Keywords: Systems Engineering; MBSE; MDAO; aircraft design

1. Introduction

The development of a new aeronautical system – an aircraft or a part of it e.g. a subsystem or a component – is becoming increasingly challenging, with impact on the development time, quality and cost. Three major challenges can be identified. The first challenge concerns the increased complexity of the system development process. This complexity is reflected in the higher quantity of generated and shared data and information, which may be often uncertain or largely unknown. This trend follows the introduction of new aeronautical technologies, for example new and more efficient propulsion systems compliant with more stringent environmental constraints, e.g. reduction of CO_2 and non-CO_2 emissions in flight. Novel technologies never installed before on operative aircraft increase the uncertainty of their behavior and impact at aircraft-level. The second challenge refers to the fact that nowadays new systems are not designed and built by single organizations, but they are fruit of the collaboration among the different entities belonging to the whole aeronautical supply chain. This means that heterogeneous knowledge and expertise is distributed among multiple organizations, which have to exchange design data and information. Lastly, the life cycle of an aeronautical system spans decades from conceptual design to retirement, going through multiple stages, such as development, production, operation and support. During its life cycle, the aeronautical system under design interacts with many other systems – defined as “enabling systems” – which *facilitate the life cycle activities of the system under design but are not a direct element of the operational environment* (adapted from [1]). Therefore, in the design phase, multiple constraints and aspects belonging to the whole life cycle of the system under design have to be considered, and often traded among each other.

In order to tackle all those challenges, methodologies supporting the development of new and innovative systems are required. A methodology is composed by processes, methods and tools [2]. A *process* is defined as a *logical sequence of tasks performed to achieve a particular objective* [2]. For example, a System Life Cycle Technical Process like the one standardized by the ISO 15288 [3] can be adopted to develop a new system by performing tasks as definition of stakeholders and collection of their needs, transformation of needs into technical requirements, generation of multiple system concepts (also called *architectures*) compliant with the requirements, and verification and validation activities. The Product Development process like the one standardized by the ISO 15288 can be adopted in a Systems Engineering approach, whose ultimate goal is the generation, design

and optimization of system architectures compliant with different stakeholder needs and requirements. A *method,* instead, *specifies which techniques can be adopted to perform the tasks of a process* (adapted from [2]). During the various activities of a Systems Engineering Technical Process [1], which guides the development of a new system, a lot of information is produced. This information can be collected in multiple documents, but this hampers the system development due to several limitations. For example, the collaboration among multiple experts is negatively impacted, since the information they produce and share is spread in different texts, tables, drawings. Also, the relations between all the pieces of design information are not clear, and therefore the traceability is hindered. A model-based method instead can improve all the activities of a Systems Engineering process. Models can facilitate the collection and sharing of information, improve the traceability, and increase the opportunities for automation (e.g. for checking consistency among the entire design data) and reuse of data, hence accelerating the whole development process. This is why new methods belonging to Model-Based Systems Engineering (MBSE) approaches are becoming even more popular since the last decades among multiple practitioners [4]. MBSE is defined as the *formalized application of* ***modeling to support*** *system requirements, design, analysis, verification and validation activities throughout* ***development*** *and later* ***life cycle*** *phases* [5]. Methods used in MBSE approaches support activities regarding the identification of stakeholders, collection of their needs, transformation of needs into requirements and generation of system architectures. Other methods instead support the system design and optimization activities. Methods for Multidisciplinary Design Analysis and Optimization (MDAO) are in fact conceived and adopted to accelerate and enhance the collaboration among the different disciplinary experts who provide their disciplinary analyses (e.g. aerodynamics, structures) necessary to design and optimize a new system. Finally, a tool is defined as a *means implementing the techniques to perform the tasks of a process* (adapted from [2]). A tool can simply consist of pen and paper, or, nowadays more commonly, software and codes, implementing model-based methods.

Several methodologies that leverage an MBSE approach and support the development of systems are available in literature. A survey of some among the most notable MBSE methodologies is provided in [2]. Many of these methodologies include methods prescribing how to model and represent the information produced during the development process (e.g. ARCADIA and its implementation into Capella [6], or architectural frameworks as Zachman's Framework [7], DoDAF [8], MODAF [9], NAF [10] and TOGAF [11]). Others methods are proprietary and no information is publicly available (e.g. Airbus' MOFLT method [12] and its implementation into the software CAMEO System Modeler). In general, all these MBSE methodologies are effective, although some activities addressed by them (e.g. requirements engineering, system architecting) might be improved by integrating and extending other methods available in literature.

Since several years, the Institute of System Architectures in Aeronautics of the German Aerospace Center (DLR) is performing research addressing the investigation and developments of new processes, methods and tools, by making use and extending the best approaches available in literature while overcoming the current limitations. The processes, methods and tools investigated or developed by DLR are briefly described in the paper, while more detailed information is provided in the references. After an introduction in section 2 of the development process set up at DLR, section 3 presents processes, methods and tools supporting the definition of stakeholders, needs and requirements. Section 4 instead focuses on system architecting activities. Processes, methods and tools supporting design and optimization activities are described in section 5, while section 6 addresses trade-off and decision-making activities. Finally, impacts and benefits of the proposed processes, methods and tools, but also limitations and future research activities, are listed in the conclusive section 7.

2. Process for the definition, architecting, design and optimization of systems

The processes, methods and tools tackled at DLR support the main system development activities included in the overall development process schematized in Figure 1. These activities are grouped in three phases:

1. ***System Definition* phase**, which includes the main initial activities of a typical (document or model-based) Systems Engineering Product Development process: identification of system stakeholders, collection of their needs about the system, and translation of these needs into technical requirements. This phase covers the blocks "*System Identification*" and "*System Specification*" of Figure 1, and it is addressed in section 3;
2. ***System Architecting* phase**, which includes other main activities of a typical Systems Engineering Product Development process addressing the generation of the various and alternative architectures (different configurations) of a system. This phase covers the blocks "*System Architecting*" and "*System Synthesis*" of Figure 1, and it is addressed in section 4;
3. ***Design and Optimization* phase**, which aims at identifying and selecting disciplinary competence (e.g. aerodynamics), formulating and executing MDAO processes and exploring and assessing the solutions space. This phase covers the block "*System Exploration*" of Figure 1, and it is addressed in section 5.

Additionally, this process includes a fourth phase, which connects system definition activities with design and optimization: the ***trade-off and decision-making* phase**. The goal here is to explore various solutions – characterized by different architectures and specifications – and evaluate them on the basis of the different stakeholder needs. Trade-off and decision-making activities are done in this phase, as described in section 6.

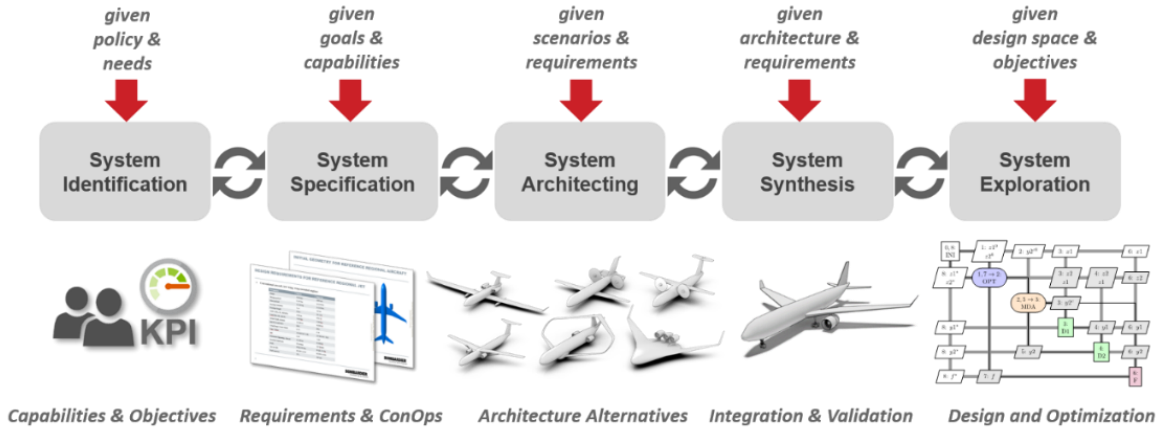

Figure 1 – Schema representing the proposed Systems Engineering development process (adapted from [13]).

The processes, methods and tools that are addressed in this work are conceived to support the development of conventional systems (e.g. those that are characterized by traditional configurations and technologies), but also innovative ones (*Challenge #1*). Indeed, the determination of a specific solution, either conventional or innovative, happens in the "*System Architecting*" block of Figure 1. Before the determination of a specific solution, the development activities target the definition of the system in terms of its expected functionalities and behavior, independently of the technologies adopted. This means that the proposed process is built in a way to let the designers identify and consider multiple (innovative) solutions, without being biased since the beginning by a specific solution characterized by specific technologies. Additionally, the methods that support the development process are developed with the aim of enhancing the collaboration among the multiple experts involved during the development activities (*Challenge #2*). In this regard, the transition from document-based to model-based methods aims at improving the sharing of data and information among the different designers. Finally, it needs to be clarified what the "system" developed through the processes, methods and tools can be. It can be an aircraft or part of it (e.g. a subsystem or a component). However, the developed "system" can be an enabling system that supports the aircraft during its life-cycle. For example, a manufacturing system composed by different manufacturing machines and tools have to be developed and built to support the aircraft during its production stage. Similarly, a maintenance system can be assembled and operated during the aircraft's operation stage. Hence, the proposed processes, methods and tools should support the simultaneous development of more systems interacting with the aircraft during its whole life-cycle (*Challenge #3*).

3. Processes, methods and tools for the *System Definition* phase

The *System Definition* phase aims at defining system technical requirements translating the various stakeholder needs. This phase of the development process starts with the identification of the system stakeholders. 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***" [3]. When considering an aircraft as a *system*, examples of stakeholders might include: airlines, passengers, cabin and ground crews and maintainers. The multiple stakeholders express different ***needs***, which are "***informal expressions of something that has to be provided, ensured or avoided by a system or the development project of this system***" [14]. 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, general, or ambiguous terms, they must be used to originate *requirements*, which are characterized as unambiguous, complete, feasible, verifiable and correct. A ***requirement*** is defined as "***a statement which translates or expresses a need and its associated constraints and conditions***" [14]. Different types of requirements can be identified, but more generically requirements can be divided between *functional* or *non-functional requirements*. *Functional requirements* express the *functions* that the system should perform [15]. A ***function*** is *what* a system does [16]. *Non-functional requirements,* instead, specify parameters that can constrain the designed solution. Among all the non-functional requirements, the performance ones define at what level the system has to perform the requested functions [15]. Other types of non-functional requirements can be found in [17].

Properly understanding what the stakeholders want from a system, and correctly translating these needs into requirements, have an important role in the design of a system. For this reason, a structured approach guiding the complete and clear collection of stakeholder needs and their transformation into requirements is recommended by the authors (more information can be found in [17]), other than by multiple organizations. The International Standard ISO/IEC/IEEE 29148 [14] provides standard guidelines addressing requirements engineering processes. This is also endorsed by the International Council on Systems Engineering (INCOSE), which established a Requirements Working Group that in 2017 published a guide addressing recommendations and rules for writing good requirements [18], so requirements can be correct, complete, unambiguous and verifiable. The requirement type mentioned before prescribes how the requirements text has to be written. In fact, in order to be complete, requirements should follow a predetermined structure - called pattern – that prescribes both mandatory and optional text elements (the latter included in square brackets), as shown in Figure 2. Additionally, the text of the requirements should comply to several grammatical and syntactical rules, which are collected in [18].

Functional requirements:
Pattern: The SYSTEM shall [exhibit] FUNCTION [while in CONDITION]
Example: "The aircraft shall provide propulsive power [during the entire mission]"

Performance requirements:
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"

Figure 2 – Requirement patterns depending on requirement types (adapted from [17])

Additionally, a series of attributes characterizes each requirement. Attributes are additional elements used to support the management of the requirements. Example of requirements are: identification number (ID), parent source (the origin of a requirement), Means of Compliance (means used to verify that the designed system is compliant with a specific requirement) and the requirement version. A long list of attributes is recommended by INCOSE [18].

All the information relative to the stakeholders and needs, and regarding the requirements, expressed both through the requirements text and the attributes, can be collected either in documents (traditional approach) or in models (as part of an innovative MBSE approach). The development methodology proposes the SysML modeling language [19] to represent the models of stakeholders, needs and requirements. Although SysML is recommended by INCOSE for MBSE activities [1], it is recognized by the authors 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 and described in [17]. As an example, the SysML diagram of Figure 3 can be used as a template to model the text of a performance requirement, which is characterized by mandatory and optional elements as *system*, *function*, *performance*.

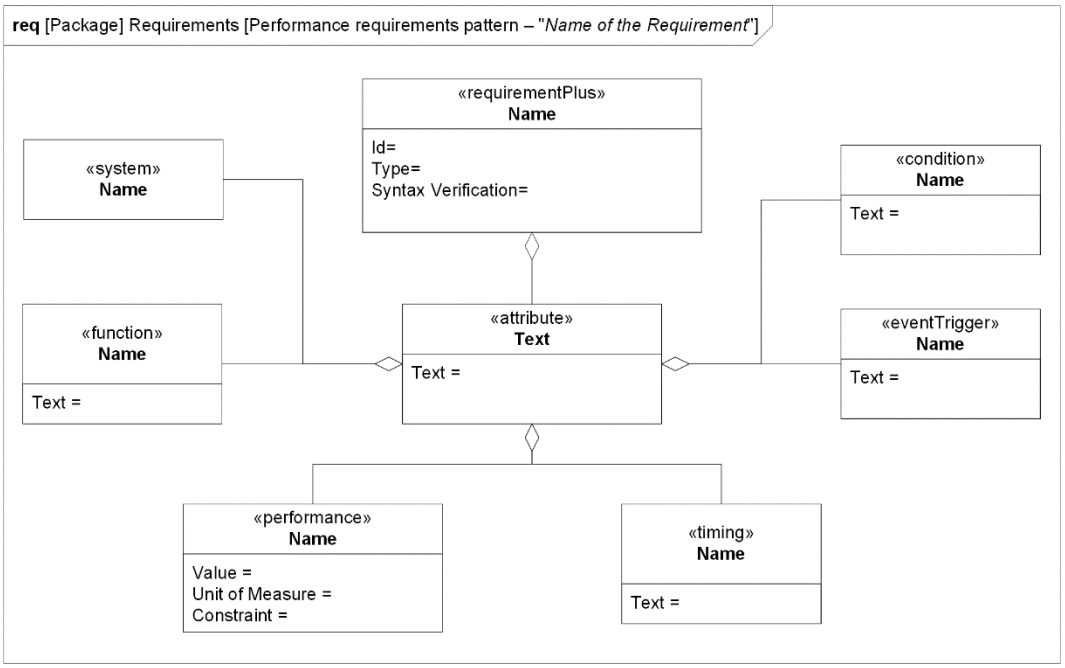

Figure 3 – SysML Requirement Diagram representing the template prescribed in [17] to model performance requirement

An example of application of requirement model built by following the DLR's structured approach is shown in Figure 4. The example of performance requirement of Figure 2 is modelled according to its relative patterns. Therefore, its text "*The aircraft shall fly at min Mach 0.8 during cruise*" is represented by the different elements "system" (i.e. the aircraft), "function" (i.e. to fly), "performance" (i.e. Mach) and "condition" (i.e. cruise)

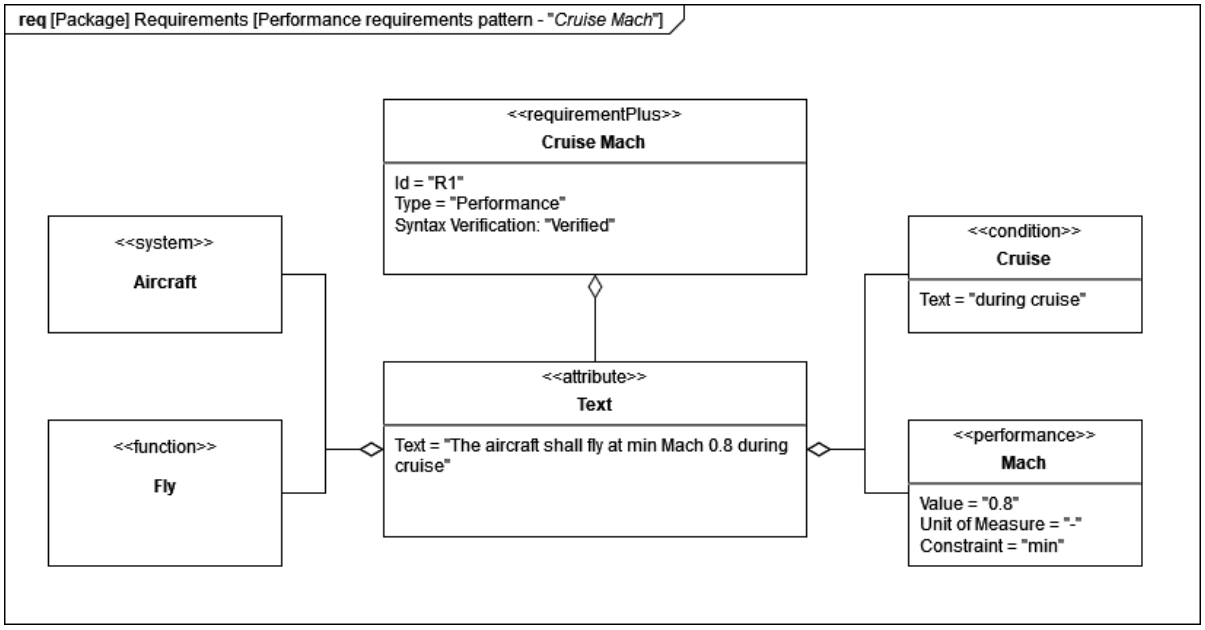

Figure 4 – SysML Requirement Diagram representing the template prescribed in [17] to model performance requirement

Several tools supporting the definition, modeling and management of stakeholders, needs and requirements are available in literature, both commercially and with free access. One of the most commonly used tools for requirements management is IBM Doors. Although this tool is quite

extensively adopted in industrial contexts, it is still document-based, and it doesn't support the designers with patterns and rules compliancy. Other tools instead implement a model-based approach, e.g. Papyrus, IBM Rhapsody, CAMEO System Modeler. These tools allow the traceability with other system elements (e.g. functions and components), but the modeling of requirements by following a structures approach is still an almost full manual task performed by the users. To overcome these limitations, a new tool named ARMADE [20], is being developed by DLR. It is web-based to allow the collaborative definition and management of stakeholders, needs and requirements, and it supports the modeling of requirements according to the method described in this paper. ARMADE implements the patterns described in [17], but it allows also the modeling of requirements according to any structure decided by the users. This gives even more flexibility to this part of the development process, making possible to adopt the tool in different programs involving different organizations using their own structure.

4. Processes, methods and tools for the *System Architecting* phase

The next phase of the proposed development process aims at defining and modeling functional, logical and physical system architectures. Functions are extracted from functional requirements and used to generate ***functional architectures***, which represent all the functions that the system should fulfil, regardless of its implementation, i.e. independently from what components of the system will fulfil that function. Indeed, it is aim of the ***logical architecting*** phase of the methodology to identify solution-neutral components (called ***logical components***) that can fulfil the functions. Logical architectures are finally transformed into ***physical*** ones, by identifying instances of the previously determined logical components.

A novel process for the generation of system architectures is accurately described in [21] and [22]. This process is divided in three parts: functional, logical and physical architecting. The functional system architecting process aims at identifying all the functions that the system should perform. These functions are called boundary functions and derive from the previously determined functional requirements. These functions focus on a single *level of elaboration* of the system under development, e.g. aircraft or propulsion system levels. For example, if the focus is on an aircraft level, boundary functions might encompass: *to transport payload*, *to generate lift* and *to provide thrust*. Alternatively, the focus might be on a fuel system, and the examples of boundary function might be to store fuel and to transport fuel. Once boundary functions have been determined, they have to be allocated to logical components. For example, the aircraft-level function *provide thrust* can be allocated to the logical component *engine*. This part of the architecting process is called logical architecting. Additional functions can be identified even during this part of the process, as components might induce other functions (called *induced functions*) that, in turn, are fulfilled by other components. For example, an engine would induce the function *provide fuel*, which would be allocated to a *fuel system*. Finally, the physical architecting part of the system architecting process identifies suitable physical components compliant with the non-functional requirements. A physical component instantiates a logical one by specifying characteristics as performance, dimension, material attributes. For example, the *CFM56* is a physical component characterized by certain mass, thrust, fuel consumption, which instantiates the logical component *engine*.

Several methods are proposed in literature to support a system architecting process. Some methods focus on the definition of functions from functional requirements, e.g. FAR (Functional Architecture by use case Realizations) [23]. Other methods address the functional architecting process, like the FAS (Functional Architectures for Systems) [24]. Other methods can be adopted to support the whole process, e.g. SYSMOD [25]. All these methods support the development of single architectures. The methods created by the DLR research group to which the authors belong instead supports the modeling of the *Architectural Design Space* [26]. This design space defines all the possible components that can be part of an architecture given a list of boundary functions. This means that more than one alternative component can be allocated to a given function. For example, the function *generate thrust* can be allocated to a *turbojet engine* or a *turbofan engine*. Therefore, an *Architectural Design Space* represents multiple system architectures, which can be both conventional and innovative. The main advantage of this approach lies in the fact the system architect is not biased by specific (often conventional) solutions, but many more unconventional but promising solutions can be conceived as well, perhaps resulting in the end better than the other ones.

The proposed method employs the modeling language called *Architectural Design Space Graph* (ADSG) for the representation of the various alternative architectures. An example of an ADSG representing various hybrid-electric propulsion system architectures is depicted in Figure 5 [27]. The graph includes several modeling elements, which are described in [28]. Among all of them, the elements *FUN* represent both boundary and induced functions, while the elements *COMP* represent system components, which are connected to the functions through two different types of connections. The connection *fulfilled by* represents the allocation of a function to a component. The connection *needs* represents the induction of a function. In the specific example, the hybrid-electric propulsion system under development has to fulfil the function “Provide Propulsion”, which can be done by a “Propeller” component that needs the induced function “Gen(erate) Power”. This function can be fulfilled by an electric motor (“Motor” component), or by a turboshaft (“Mech Turboshaft” component), or by both. During the generation of single architectures, the selection of one of these components entails the need of additional components fulfilling induced functions. For example, if the electric motor is selected, all the components relative to the electric drive (e.g. DC Bus, Inverter, Batteries) are required. Analogously, when the turboshaft is chosen to be part of the architecture, a fuel system shall be included as well.

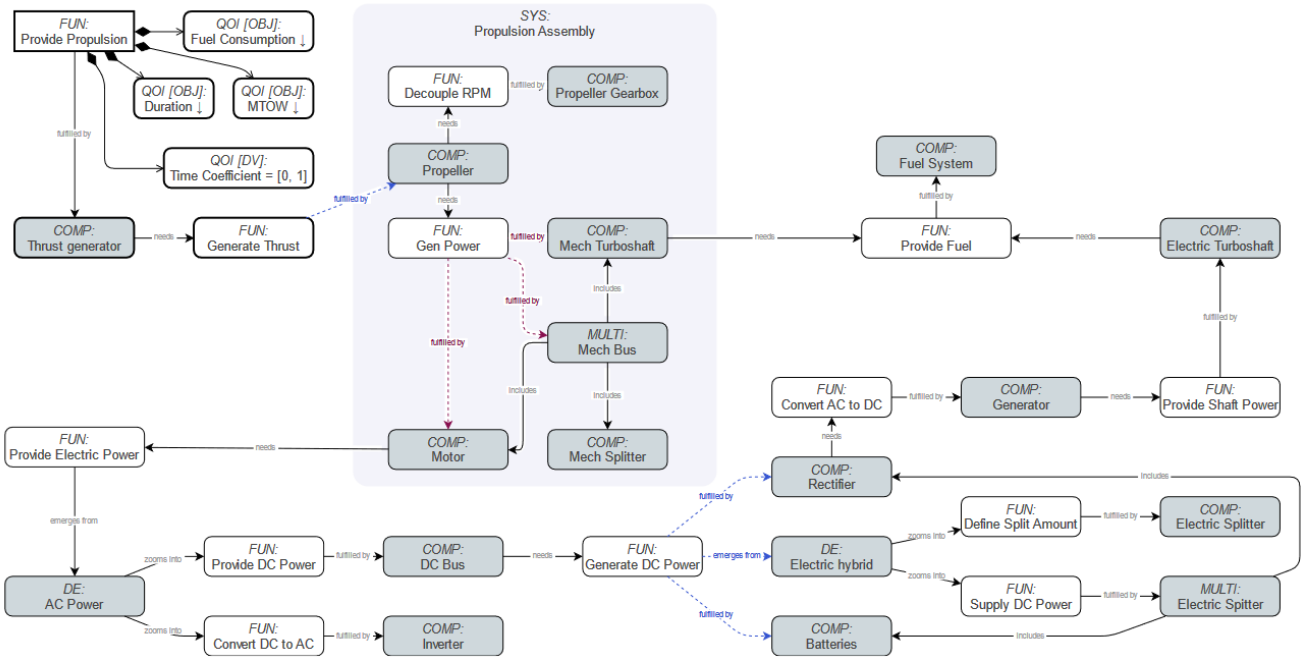

Figure 5 – ADSG representing various architectures of a hybrid-electric propulsion system [27]

Eventually, the proposed system architecting method aims at generating single instances of system architectures. This can be done by taking architectural decisions, i.e. by deciding which components among all the possible alternatives can fulfil the system functions. Generating architectures can be a manual task, when the system architect manually selects the components being part of the architecture, or an automated one, when architectural decisions are taken by an algorithm. The latter case can belong to a *System Architecture Optimization* (SAO) problem, in which the alternative architectures are part of the optimization problem together with other design variables. This method entails the integration of system architecting into an MDAO approach (which is addressed in section 5), which accelerates the assessment of multiple alternative solutions. More details about SAO can be found in [29] and [22], whereas research about optimization algorithms for system architectures is published in [30], and [31].

Many tools are being developed by commercial parties and by the research community to support system architecting activities. The majority of these tools (e.g. CAPELLA, IBM Rhapsody, CAMEO system modeler) can be used to model single architectures in a standard modeling language (e.g. SysML). A web-based tool developed by DLR is named ADORE [29], which supports the system architect in modeling the Architectural Design Space and (automatically) generating system architectures by taking architectural decisions. ADORE can be included within an MDAO toolchain to enable architectures optimization, as described in the following section. To the authors knowledge,

there are no other tools in literature supporting the modeling of the Architectural Design Space, the automatic generation of architectures, and the inclusion of system architecting into an MDAO process.

5. Processes, methods and tools for the *Design and Optimization* phase

The transition from logical to physical architectures, i.e. the instantiation of components through their characterization in terms of specifications as mass, performance and dimensions, can be done by exploiting an MDAO approach. The *Design and Optimization* phase of the proposed development methodology aims at supporting the formulation and execution of MDAO workflows for the design and optimization of all the previously generated architectures, ensuring that they comply with the non-functional requirements. Therefore, this phase of the development methodology entails a change of perspective. In the previous phase, the perspective was *architectural*, since the focus of the development methodology was on the characterization of system's components, e.g. fuselage, wing, engine. In this phase, the perspective becomes *disciplinary*, since the focus moves from the architectural components to all the engineering disciplines (e.g. aerodynamics, structures, flight mechanics) needed to design and optimize the generated system architectures. It is therefore aim of this second phase to formulate multidisciplinary workflows (e.g. identify the required disciplines) and execute them (i.e. to execute toolchains consisting of tools implementing the various disciplines).

Several methodologies are being developed since years to support the formulation and execution of MDAO workflows. Ciampa *et al.* [32] have seen an evolution of these methodologies in time, and they have identified three generations of MDAO, characterized by increasing level of distribution of tools and collaboration between disciplinary experts. The interest of the authors lies in the methodologies belonging to the last generation, which support and enhance the collaboration between all the disciplinary experts involved in the MDAO, by overcoming several non-technical barriers identified in [33]. In general, disciplinary experts tend to focus on their individual disciplines, with the risk of missing the overview of the complete MDAO problem and the interactions with all the other disciplines. A process and methodology for the acceleration of MDAO formulation and execution was developed by DLR in the past and is called the AGILE paradigm [34]. This process and methodology are still in use and under further development by the research group to which the authors belong [35], [36]. An example of current development regards the dynamic formulation of MDAO workflows, which are automatically adapted to the specific system architecture proposed by the optimizer at each iteration of a SAO problem. More details can be found in [37].

The methodology proposed by the research group can be divided into two parts. The first aims at formulating and modeling an MDAO problem in the form of an XDSM (eXtended Design Structure Matrix). This formulation was introduced by [38], and it consists of a graphical representation of the MDAO problem in terms of the involved disciplines, the connections (in term of exchanged inputs - outputs) between disciplines, and the resolution strategies, e.g. converged analysis, optimization, Design of Experiment (DOE). An example of XDSM representing an optimization problem of a hybrid-electric powered aircraft is depicted in Figure 6 [35]. The process for the MDAO formulation and its modeling through an XDSM starts with the identification and collection of disciplines required to solve a particular MDAO problem, in this case "Performance", "Engine", "Hybrid System", "Mission" and "Masses". Every discipline is characterized by several inputs and outputs. Knowing these input and outputs allows to establish the connections between the different disciplines. For example, the "Engine" discipline estimates the Specific Fuel Consumption (SFC), which is updated by the "Hybrid System" discipline, before being used by "Mission" to compute the mission fuel mass. These connections can be easily realized when the various disciplines adhere to a common data schema, e.g. a standard structure of all the variables (e.g. masses, geometries, dimensions) relative to the system being designed. The CPACS standard is one example of a common data schema [39]; CPACS was developed by the DLR and it is used by several organizations (e.g. [40]). A first version of an XDSM can be built represent all the required disciplines and their connections. Further activities are then performed, resulting in an update of the XDSM. These activities include:

- **Order the disciplines**: several strategies can be adopted to order the disciplines and contribute to accelerate the execution of design and optimization. For example, one strategy would be the minimization of feedback loops, which happen when a variable needed by a discipline is computed by another discipline executed later during the MDAO problem.

- **Inspect the connections and resolve collisions**, which happen, for example, when multiple disciplines update the same variable.

- **Architect the MDAO problem**, defining convergence loops according to different strategies (e.g. Gauss-Seidel and Jacobi), formulating optimizations by determining design variables, constraints and optimization objectives, and defining the various design experiments of a DOE.

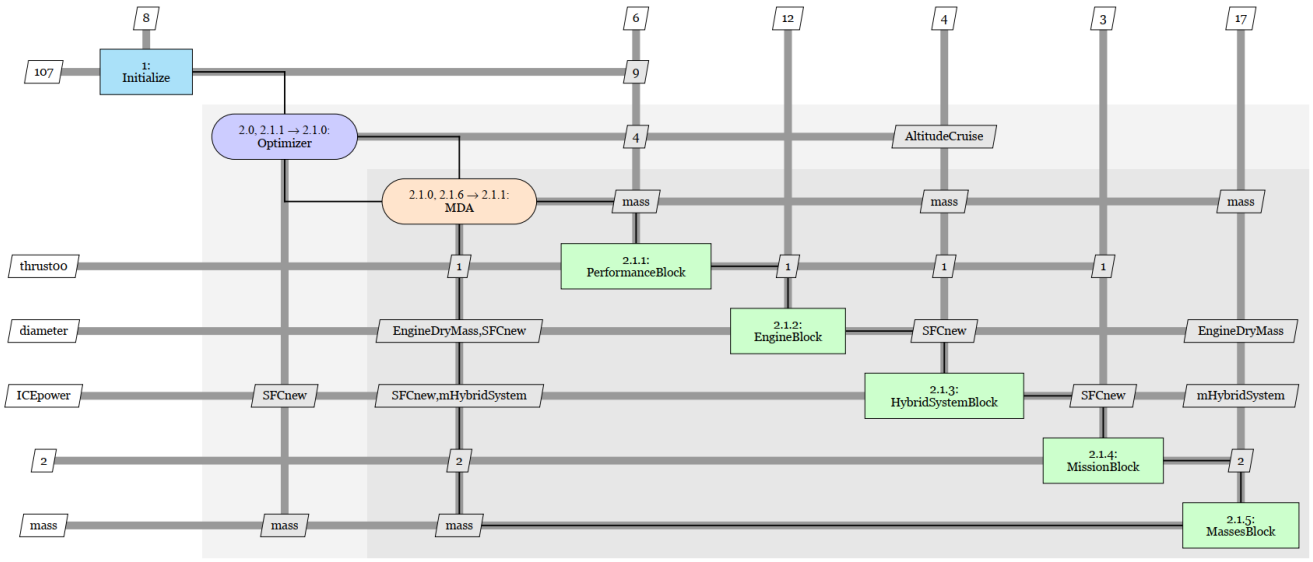

Figure 6 – XDSM representing an MDAO problem for the optimization of a hybrid-electric powered aircraft [35]

The MDAO problem previously formulated and represented in an XDSM format consists of a blueprint of the MDAO workflow that has to be implemented and then executed. The second part of the proposed MDAO method focuses the execution of the MDAO problem. This means implementing all the disciplines with executable disciplinary tools. These tools might be owned by different departments or organizations, and be implemented through different coding languages. Therefore, all the tools belonging to a common MDAO workflow have to adhere to the common data model, and should be executed without any user interaction, in order to maximize the level of automation. In addition, the different disciplinary tools belonging to the different organizations might be located in different places, for intellectual property reasons. Methods and tools are required to support the execution of MDAO workflows with disciplinary tools spread in multiple locations, as described in [41].

Many modifications might affect the formulation and execution of an MDAO problem, e.g. the addition or removal of a discipline. Therefore, the methodology should allow the flexibility needed to quickly account for any change. In addition, several activities might require user intervention, but they might be accelerated in case automation would be leveraged. Finally, continuous verification of the MDAO problem (e.g. check that variables are correctly exchanged between disciplinary tools) and documentation are needed to ensure the correct formulation and execution of the design and optimization process. Therefore, several commercial and research tools are being developed to support, accelerate and improve MDAO activities. For example, among the tools developed by academic and research organizations it is worth mentioning TU Delft's KADMOS [42], NASA's OpenMDAO [43], ONERA's WhatsOpt [44] and IRT's GEMSEO [45]. Two MDAO tools are also developed by DLR: one – named MDAx [35] - supporting the formulation of collaborative MDAO workflows, and one – named RCE [46] – supporting the execution of MDAO workflows. The collaboration in the MDAO processes between various disciplinary experts is enabled by MDAx by supporting the formulation of workflows based on CPACS or other standard schemas for the collaborative exchange of aircraft design data. Another main feature of MDAx is the automatic generation of ready-to-be-executed workflows, which can be imported in OpenMDAO, RCE, or into other MDAO tools. Additionally, a continuous verification of the XDSM model built through the interactive Graphical User Interface (GUI) of MDAx reduces the risk of making design errors, hence reducing the time required to formulate and execute an MDAO workflow.

6. Value-Driven trade-off and decision-making

All the processes, methods and tools previously described can be leveraged for the conceptual design of a new aircraft (or part of it), mainly with focus on the operation stage of its life-cycle. For example, the aircraft might be developed in order to maximize its flight performance, e.g. minimizing the fuel consumption. However, additional life-cycle stages have to be considered, such as production and support (e.g. maintenance). During these stages, other systems (i.e. the *enabling systems* introduced in section 1) interact with the new aircraft. The methodology addressed in this paper can be adopted to develop enabling systems, too. In other words, the same processes, methods and tools can be used to define stakeholders, needs, requirements, functions, components and specifications of the enabling systems. An example of enabling system is an aeronautical supply chain, which is defined as a combination of multiple organization spread in different locations and with the various competencies required to produce an aircraft, its subsystems and its components.

In most cases, the aircraft is developed first, and after all its enabling systems. This for example can happen with the development of the supply chain and manufacturing systems (i.e. consisting of different manufacturing machines and tools). In fact, architectural and design decision addressing these enabling systems are taken only once the aircraft architecture has been frozen [47]. Although this approach might entail an optimal aircraft solution from its operational perspective, this solution might not be compatible with production constraints (in term of feasibility of realizing the aircraft). As a consequence, the aircraft would have to be redesigned, hence entailing an increase of development costs [48].

Therefore, a different approach is investigated by the authors (described in details in [49] and [50]), in which the development process addresses simultaneously multiple systems (i.e. the aircraft and the enabling systems) and constraints belonging to multiple life-cycle stages. This means that all the life-cycle stages (not exclusively the operational one) are accounted during the development of the aircraft, and other enabling systems are developed since the beginning of the whole process. Although this approach entails the advantage of determining the “best” solution, which is the one not focusing on a single system (e.g. the aircraft) and addressing only operational constraints, a new challenge arises. This challenge lies in the formalization and modeling of the existing complex interconnections between the various systems already at the beginning of the aircraft life-cycle (particularly at the conceptual design stage), before the production of the aircraft starts.

In order to simultaneously develop multiple systems, accounting in the design for the various constraints belonging to the different life-cycle stages, the proposed methodology is based on the value-model theory. In the specific, this theory is leveraged to weight and combine together (into a parameter called *value*) various constraints belonging to different systems and different life-cycle stages, constraints that are derived from different stakeholder needs. The equation (1) is taken from the Multi-Attribute Utility (MAU) value model [51] and is used to compute the *value*, a number ranging between its 0 and 1, where 1 identifies the “best” solution accounting for the different systems and constraints. Since various architectural and design decisions can be taken for each system addressed in the development, and different weights can be assigned to the various constraints, multiple solutions can be generated and evaluated, each one characterized but its own *value*. A trade-space with various solutions and different *values* can therefore be explored and assessed, supporting the decision-makers in taking decisions to eventually select the “best” solution.

$$value = \sum_{i=1}^{N} \lambda_i U(X_i) \tag{1}$$

where:

- N is the number of constraints belonging to various systems and life-cycle stages, and derived from different stakeholder needs;

- X_i is a specific constraint, e.g. fuel consumption (constraint belonging to the operational life-cycle stage of an aircraft) or production quantity (constraint belonging the production life-cycle phase of an aircraft and impacting on manufacturing system and supply chain);

- $U(X_i)$ is the single-attribute utility function, which is used to quantify decision-makers' expectations with respect to each constraint. Each of these functions represents the way decision-makers would select a solution by only considering the specific constraints. These functions are used to translate the qualitative decision-makers' preferences into analytical curves, like the one of Figure 7. The example in the figure shows how much the decision-makers are "satisfied" (this is measured with the dimensionless parameter *utility* of the y-axis) by the *time* (represented in the x-axis, normalized) that the production stage might require. Higher is the production time, lower is the satisfaction of the decision-makers, and therefore lower is the resulting *utility* parameter.

- λ_i is the weight associated to each constraint X_i. The weights represent the relative importance of each constraint, since the decision-makers might want to give more emphasis on e.g. the aircraft performance, or the production cost, or the production quality.

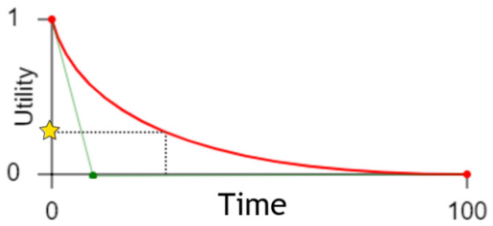

Figure 7 – Example of a single-attribute utility function quantifying the decision-makers' expectation with respects of the production time parameter [49]

After having architected and designed different alternatives of systems (aircraft and enabling systems), and for each solution having determined its *value*, a trade-space exploration can be performed. Figure 8 shows an example of trade-space. The graph on the center-right side represents 19 possible solutions, all of them representing a single designed aircraft, but manufactured by different manufacturing systems and supply chains. Every solution is assessed in terms of production cost (which is represented by the normalized parameter of the x-axis) and of *value* (which is compute with equation (1), built with the single-attribute utility functions represented on the left side of Figure 8, and reported on the y-axis). The resulting trade-space can be analyzed by the decision-makers when selecting the "best" solution. In this specific case, solution #1 is the one characterized by the highest *value*, although the decision-makers may opt for other solutions (e.g. solution #19), with slightly lower *value* but significant lower production costs.

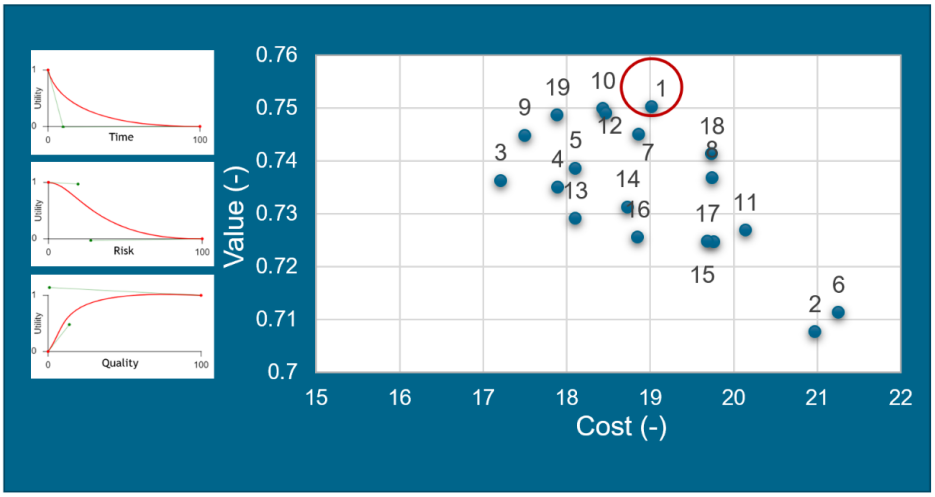

Figure 8 – Example of a value-driven trade-space representing multiple design solutions with different production cost and *value* [49]

This methodology allows decision-makers in analyzing multiple scenarios, where single-attribute utility functions and weights can be changed. To the authors knowledge, there are not tools in literature implementing the described process and method for value-driven trade-off and decision-making. Therefore, the authors have developed the in-house tool VALORISE [49], which interactively supports the decision-makers in identifying constraints, building the different utility functions, and visualize the trade-spaces by changing weights.

7. Conclusions

A new methodology supporting the development of aeronautical systems addressing their entire life-cycle has been presented in this paper. The paper aimed at providing an overview of the processes, methods and tools addressed by the research group where the authors belong to. More details are described in the references reported in the different sections. Multiple are the advantages of this methodology. These advantages – which will be quantitatively assessed in future – include:

- **Increased traceability between design/architectural data and decisions**, since they are part of models (e.g. requirements model, architectures model) that are connected together (e.g. requirements derive functions, which are allocated to components of an architecture).

- **Complete collection of all stakeholder needs and derived requirements**, since a Systems Engineering approach gives particular emphasis on this part of the system development process.

- **Automatic verification of the models**, identifying soon potential errors that might happen during the development process, hence minimizing the risk of time-consuming redesigns.

- **Increased exploration of the design space**, which can include many more innovative architectures that can be designed and optimized.

- **Acceleration and improvement of the development process** (e.g. the formulation of an executable MDAO workflow), mainly by automating (repetitive) activities.

- **Enhanced collaboration between multiple experts**, since they produce/collect information from/into models rather than documents.

- **Re-use of models in other projects or programs**, hence reducing the development time.

- **Simultaneous modelling of multiple systems**, hence allowing the tradeoff of different solutions that include constraints belonging to the whole life-cycle.

- **Reduce the entry barrier for "non-MBSE experts"** thanks to the developed tools, barrier that is caused by the introduction of a new approach (i.e. a model-based one) and new modeling languages (e.g. SysML).

- **Inclusion of the experts into the design/architecting loops**, since automation can provide support, but the ultimate design and architecting decisions are taken by the experts, who are informed by the models.

The research group at DLR is still working on improving, developing and testing processes, methods and tool. In particular, future activities will address the inclusion of new methods (e.g. based on the support given by Artificial Intelligence) and new modeling languages (i.e. SysML v2). More application cases will be addressed, tackling the development of systems bellowing not only to the aeronautical domain, but also to other domains, space, transport and energy. These application cases will be talked to test the methodology, to quantify its benefits, and to identify limitations that will require further research.

8. Contact Author Email Address

E-mail: Luca.Boggero@dlr.de

Phone: +49 40 2489641-338

Address: Hein-Saß-Weg 22, 21129 Hamburg, Germany

9. Copyright Statement

References

[1] INCOSE, Systems Engineering Handbook v.5, 2023.

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

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

[4] 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.

[5] Technical Operations - INCOSE, "Systems Engineering Vision 2020 - INCOSE-TP-2004-004-02," 2007.

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

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

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

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

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

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

[12] R. H. Madeira, D. H. de Sousa Pinto and M. Forlingieri, "Variability on System Architecture using Airbus MBPLE for MOFLT Framework," in *INCOSE International Symposium*, Honolulu (US-HI), 2023.

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

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

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

[17] 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.

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

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

[20] A. Chojnacki, G. Donelli, L. Boggero, P. Shiva Prakasha and B. Nagel, "Adaptable MBSE Problem Definition with ARMADE: Perspectives from Firefighting and AAM SoS Environments," in *EASN*, Thessaloniki (GR), 2024 (to be published).

[21] L. Boggero, P. D. Ciampa and B. Nagel, "An MBSE Architectural Framework for the Agile Defintion of Complex System Architectures," in *AIAA Aviation Forum*, Chicago (US-IL), 2022.

[22] J. Bussemaker, L. Boggero and B. Nagel, "System Architecture Design Space Exploration: Integration with Computational Environments and Efficient Optimization," in *AIAA Aviation*, Las Vegas (US - NV), 2024.

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

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

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

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

[27] J. Bussemaker, R. García Sánchez, M. Fouda, L. Boggero and B. Nagel, “Function-Based Architecture Optimization: An Application to Hybrid-Electric Propulsion Systems,” in *INCOSE IS*, Honolulu (US - HI), 2023.

[28] J. H. Bussemaker and P. D. Ciampa, "MBSE in Architecture Design Space Exploration," in *Handbook of Model-Based Systems Engineering*, Springer, 2023.

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

[30] J. Bussemaker, "SBArchOpt: Surrogate-Based Architecture Optimization," *Journal of Open Source Software,* vol. 8, no. 89, 2023.

[31] J. Bussemaker, P. Saves, N. Bartoli, T. Lefebvre and B. Nagel, "Surrogate-Based Optimization of System Architectures Subject to Hidden Constraints," in *AIAA Aviation*, Las Vegas (US - NV), 2024.

[32] P. D. Ciampa and B. Nagel, "Towards the 3rd generation MDO collaborative environment," in *ICAS*, Daejeon (KR), 2016.

[33] G. Belie, "Non-technical barriers to multidisciplinary optimization in the aerospace industry," in *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2002.

[34] 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.

[35] A. Page Risueño, J. Bussemaker, P. D. Ciampa and B. Nagel, "MDAx: Agile Generation of Collaborative MDAO Workflows for Complex Systems," in *AIAA Aviation Forum*, Virtual event, 2020.

[36] S. Garg, J. Bussemaker, L. Boggero and B. Nagel, "MDAx: Enhancements in a collaborative MDAO workflow formulation tool," in *ICAS*, Florence (IT), 2024 (to be accepted).

[37] S. Garg, R. García Sanchez, J. Bussemaker, L. Boggero and B. Nagel, "Dynamic Formulation and Execution of MDAO Workflows for Architecture Optimization," in *AIAA Aviation*, Las Vegas (US - NV), 2024.

[38] J. R. Martins and A. B. Lambe, "Multidisciplinary Design Optimization: A Survey of Architectures," *AIAA journal,* vol. 51, no. 9, pp. 2049-2075, 2013.

[39] M. Alder, E. Moerland, J. Jepsen and B. Nagel, "Recent advances in establishing a common language for aircraft design with CPACS," in *Aerospace Europe Conference*, Bordeaux (FR), 2020.

[40] S. Garg, P. Shiva Prakasha, A. Silva, P. Albuquerque, E. Nguyen Van, K. Kimura, Y. Sugioka and M. Jacinto, "IFAR-X: collaborative engineering framework for next generation of aerospace engineers working on aircraft design," in *ICAS*, Florence (IT), 2024.

[41] P. D. Ciampa, E. Moerland, D. Seider, E. Baalbergen, R. Lombardi and R. D'Ippolito, "A Collaborative Architecture supporting AGILE Design of Complex Aeronautics Products," in *AIAA Aviation*, Denver (US - CO), 2017.

[42] I. van Gent and G. La Rocca, "Formulation and integration of MDAO systems for collaborative design: A graph-based methodological approach," *Aerospace Science and Technology,* vol. 90, no. 4, pp. 607-627, 2018.

[43] J. S. Gray, J. T. Hwang, J. R. Martins, K. T. Moore and B. A. Naylor, "OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization," *Structural and Multidisciplinary Optimization,* vol. 59, pp. 1075-1104, 2019.

[44] R. Lafage, S. Defoort and T. Lefebvre, "WhatsOpt: a web application for multidisciplinary design analysis and optimization," in *AIAA Aviation*, Dallas (US - TX), 2019.

[45] F. Gallard, C. Vanaret, D. Guénot, V. Gachelin, R. Lafage, B. Pauwels, M. De Lozzo and A. Gazaix, "GEMS: A Python library for automation of multidisciplinary design optimization process generation," in *AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2018.

[46] B. Boden, J. Flink, N. Först, R. Mischke, K. Schaffert, A. Weinert, A. Wohlan and A. Schreiber, "RCE: an integration environment for engineering and science," *SoftwareX,* vol. 15, 2021.

[47] N. M. Gokhan, K. L. Needy, B. A. Norman and B. Hunsaker, "Benefits of incorporating supply chain decisions into the product design via design for supply chain," in *IIE Annual Conference. Proceedings. Institute of Industrial and Systems Engineers (IISE)*, 2008.

[48] O. Labbi, L. Ouzizi and M. Douimi, "Simultaneous design of a product and its supply chain integrating reverse logistic operations: An optimization model," in *Xème Conférence Internationale: Conception et Production Intégrées*, Tangier (MA), 2015.

[49] G. Donelli, L. Boggero and B. Nagel, "Concurrent Value-Driven Decision-Making Process for the Aircraft, Supply Chain and Manufacturing Systems Design," *Systems,* vol. 11, no. 12, 2023.

[50] G. Donelli, P. D. Ciampa, J. M. Mello, F. I. Odaguil, A. P. Cuco and T. van der Laan, "A value-driven concurrent approach for aircraft design-manufacturing-supply chain," *Production & Manufacturing Research,* vol. 11, no. 1, 2023.

[51] R. L. Keeney and H. Raiffa, Decisions with multiple objectives: preferences and value trade-offs, Cambridge university press, 1993.