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

10.2514/6.2021-3077

Connecting Model-based Systems Engineering and Multidisciplinary Design Analysis and Optimization for Aircraft Systems Architecting – A case study within the AGILE4.0 project

Andrew Jeyaraj[1], Nikta Tabesh[2] and Susan Liscouët-Hanke[3]
Department of Mechanical, Industrial and Aerospace Engineering, Concordia University, Montreal, Canada

The aerospace industry has set ambitious targets to meet environmental goals while under pressure to develop novel and optimized aircraft configurations effectively. Multidisciplinary Design Analysis and Optimization (MDAO) are increasingly used to optimize aircraft and their systems. Model-Based System Engineering (MBSE) methods show the potential to make the design process more effective, integrate new disciplines, and capture complex certification constraints. Today, MBSE and MDAO are not connected; different methods and tools are used, not harvesting the full potential of both approaches. This paper discusses the need for improved system architecting in the aircraft conceptual design process and introduces a framework to use MBSE in connection to MDAO. In this framework, the MBSE environment compiles system information within a system architecture specification, acting as the backbone and visual support for each stage in the systems architecting process. MDAO is used for the evaluation of system architectures.

This paper presents a case study as part of the EU-funded AGILE4.0, in which the specific link between model specification in the MBSE tool Capella and a system-level MDAO workflow is explored. Overall, this paper presents a practical contribution to linking MBSE and MDAO and paves the way for better integration of MBSE into the aircraft design process, thereby enabling MBSE implementation from conceptual design onwards. Furthermore, this will enable more detailed system analysis, such as safety analysis, starting in conceptual design, based on architecture models.

I. Nomenclature

AGILE	Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts
ARCADIA	Architecture Analysis & Design Integrated Approach
ASSET	Aircraft System Sizing Estimation Tool
ASTRID	Aircraft On-board Systems Sizing and Trade-off Analysis in Initial Design
CMDOWS	Common MDAO Workflow Schema
PVMT	Property Value Management Tool
FHA	Functional Hazard Analysis
MBSE	Model-Based Systems Engineering
MDA	Multidisciplinary Analysis

[1] Graduate student, Department of Mechanical, Industrial and Aerospace Engineering, AIAA student member
[2] Graduate Student, Department of Mechanical, Industrial and Aerospace Engineering, AIAA student member
[3] Associate Professor, Department of Mechanical, Industrial and Aerospace Engineering, AIAA Member

MDAO	Multidisciplinary Design Analysis and Optimization
PIDO	Process Integration and Design Optimization
RCE	Remote Component Environment

II. Introduction

The aerospace industry has set ambitious targets to meet environmental goals while under pressure to develop novel and optimized aircraft configurations. As a result, efforts are made to increase the effectiveness and efficiency of the design process. Within, the conceptual design phase plays a critical role, particularly for unconventional aircraft or aircraft featuring novel technologies.

The industry increasingly relies on Multidisciplinary Design Analysis and Optimization (MDAO) to explore the new aircraft design concepts or optimize more conventional designs, and MDAO is now widely used during the aircraft conceptual design phase [1]–[3]. However, MDAO and its efficient and effective use is an ongoing research field, e.g., in the EU-funded project AGILE4.0[4][4]. One of the aims of AGILE4.0 is to introduce more detailed aspects, such as certification-driven constraints, into MDAO frameworks to increase the realism of trade studies.

Within traditional MDAO frameworks, the consideration of aircraft systems (such as hydraulic, electrical, flight control, avionics, or fuel systems) is traditionally limited to overall mass estimation. Recent efforts have improved the consideration of the aircraft (sub-)system architectures in MDAO frameworks [5]–[7]. The consideration of the system architecture is vital for novel aircraft concepts featuring hybrid-electric, distributed propulsion, advanced multifunctional flight control systems, or more electrical subsystem architectures. All these examples require estimating the system architecture weight and impact on aircraft fuel burn and a means to validate the architecture's feasibility in terms of safety and certification. Within, system architecting is the process of defining a system by considering its constituent components and exchanges in response to a need, as defined by the system requirements. The resulting system architecture is a conceptual model of the underlying system that specifies the system's components, the exchanges between them and describes its ability to fulfill a set of functions.

Model-Based System Engineering (MBSE), on the other hand, is increasingly used in the industry to specify complex system architectures. MBSE can improve coherency, reduce ambiguity, and facilitate collaboration in the design of systems [8]–[12] by providing a centralized source of information throughout system development. Ideally, MBSE should be introduced in conceptual design; in this phase, requirements, system architecture, and the feasibility of the aircraft are conceptualized. Simultaneously, limited information about the subsystems is available, providing a high degree of flexibility in design. Most of the critical design decisions are made during the conceptual design. Therefore, it is the ideal point for system architects to develop modeling artifacts, to capture system requirements in the form of an architecture that can then be enriched as the design progresses through the subsequent stages.

One of the key challenges in current aircraft development processes is that MDAO and MBSE are performed as two different activity streams, using different tools and requiring specific expertise. However, both deal with translating customer requirements into a feasible solution. Also, if more detailed aircraft systems architecture evaluation is introduced into future MDAO frameworks, the specification of these architectures should ideally stem from a systematic MBSE process. Furthermore, the architectures explored and evaluated in conceptual design should ideally transition to later design stages to reduce cost and rework.

To address these challenges, this paper discusses the need for more detailed systems architecting in conceptual design and the current state of MDAO and MBSE frameworks to support this task. Secondly, this paper discusses the need to link MBSE and MDAO to enhance the system architecting and presents a framework concept. Finally, the authors present a case study in the context of the AGILE4.0 project using specific MBSE and MDAO solutions.

III. The need for system architecture consideration in conceptual aircraft-level MDAO

"Systems architecting" is defined as "…activities of defining, documenting, maintaining, improving, and certifying proper implementation of an architecture.." [13], and can be applied to any scope of any system. However, in the context of this paper, the authors focus on the conceptual portion of the architecting process (defining, documenting) and on the aircraft system architecture (Air Transport Association (ATA) chapters 2x, 3x and 4x, mainly [14]), or as described in [15], also called aircraft subsystems or on-board systems in some recent literature.

[4] AGILE 4.0: Towards cyber-physical collaborative aircraft development is a EU-funded project (2019-2022), and is the continuation of the AGILE project (Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts.

A. Systems architecting considerations for MBSE and MDAO

Typically, aircraft manufacturers or academic aircraft concept studies assume an existing system architecture when they start a new conceptual design. This simplified approach is made due to the complexity of the systems architecture and often missing data. In the past, where system architectures and interfaces between systems have been rather similar from one aircraft to another, this approach was acceptable. More detail on individual subsystems is then introduced in the preliminary design stage, based on established system boundaries and interfaces.

However, more recent system architectures, characterized by more electric and all-electric systems, are highly integrated and feature cross ATA functionality. Furthermore, More Electric Aircraft (MEA) and All-Electric Aircraft (AEA) system architectures do not benefit from the historical system weight databases, heuristic rules, and other empirical models that are available for conventional system architectures [15], [16]. The integrated and multifunctional nature of modern aircraft systems introduces additional complexity that needs to be dealt with to prevent potential integration issues later in the development process, which can lead to expensive rework, schedule overruns, and certification issues.

In addition, the advent of the SAE ARP4754A [17] now prescribes following a systematic architecting process, following a systems engineering approach. Therefore, systems architecting activities need to be performed in conceptual design to ensure that the selected architecture, developed further and implemented on the aircraft, best satisfies the requirements. In this context, MBSE is currently introduced into the design process to develop a selected system architecture in detailed design [18]. However, MBSE is not easily adaptable to conceptual design activities due to the difficulty of handling many potential architectures and variants in a multi-layered, complex specification model [19], [20]. For the conceptual design, the exploration of many different system architectures, called design space exploration, is another key activity that is currently not possible in an integrated manner with the aircraft design process.

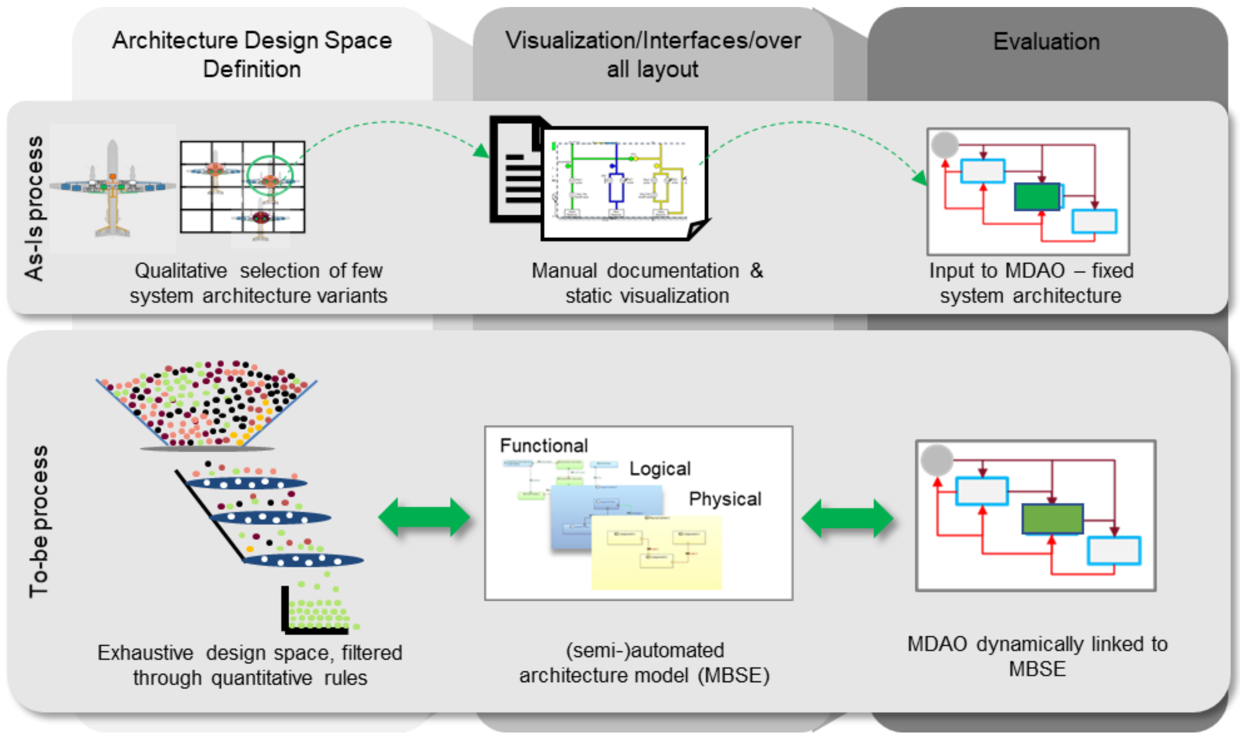
This paper focuses on system architecting activities for design space exploration during conceptual design. Fig. 1 visualizes the typical current situation in the industry (as-is process) and the envisaged future process (to-be process) for design space exploration. As shown in Fig. 1, the current process does not involve MBSE during the exploration phase. It is limited to the evaluation of few system architectures, for which the specification is manually transferred to an MDAO system-level workflow, which is then evaluated. The MBSE specification phase is disconnected and usually employed after this initial phase. However, some approaches in academia enable the evaluation of many system architectures [21]; these are typically not widely used in industry.

If considered individually, the "to-be- process" addresses the drawbacks of each step of design space exploration. The architecture definition stage lacks exhaustive consideration of all system architectures, which needs to be improved to reduce the large combinatorial design space to a smaller set of feasible architectures. To this effect, the addition of architecture safety and certification constraints are promising for identifying feasible system architectures [22].

Another aspect to consider is the nature of the system architecture description used to capture and represent information about the system configuration required to activate specific modules and interfaces in a subsequent evaluation workflow. Typical system descriptors may be textual as in [23], pictorial in [7], and object-oriented as in [24]. Based on the structuring of the architecture descriptor, it may be possible to evaluate many types of architectures, but this ultimately limits the scope for applying the architect's knowledge base. Therefore, the system architect may not be able to easily add redundancies, change power types and incorporate new technologies without having to modify a baseline workflow each time.

Presently, architecture visualization is performed using documents to categorize interfaces and present the overall system architecture layout. This approach is replaced by a model-based system architecture specification that serves to generate different visualizations and viewpoints of the system architecture. Furthermore, the model-based specification is enriched as the system architecture is further developed and can be made available to the subsystem supplier to facilitate effective system integration

To evaluate the system architecture, the conceptual designer uses sizing and analysis tools, which can be part of a system-level workflow in an aircraft-level MDAO environment. Therefore, a strategy is required in which the system-level workflow derives information from the system architecture for both configuration and execution. The system architecture is then evaluated to elicit key performance metrics, such as increments of mass, drag, fuel, and other aircraft level parameters [23], [25]. The evaluation workflow can be made generic and repeatable with minor modifications for conventional and novel system architectures [7], [23], [26]. However, reconfiguring the workflow may prove to be a time-consuming process for unconventional and novel architectures, as models need to be identified, sizing and performance routines synthesized, and a link to aircraft level metrics needs to be made.



Architecture Design Space Definition
Visualization/Interfaces/overall layout
Evaluation
As-Is process
Qualitative selection of few system architecture variants
Manual documentation & static visualization
Input to MDAO – fixed system architecture
To-be process
Functional
Logical
Physical
Exhaustive design space, filtered through quantitative rules
(semi-)automated architecture model (MBSE)
MDAO dynamically linked to MBSE

Fig. 1: Stages in the system architecting process for design space exploration

The "to-be-process" in Fig. 1 links all three steps and ensures that the information generated in the definition stage is compatible with the model-based visualization. More importantly, the MBSE-MDAO link ensures that the wealth of information about the system architecture contained in the model-based specification is accessible to the MDAO workflow. In this new process, the MDAO workflow is configurable based on the information contained within the model-based specification. Additionally, the model-based specification also provides key tool inputs that are assigned to model elements for different tools in the system-level workflow.

Thus the MBSE-MDAO link forms a crucial part of the updated system architecting process. However, there are some challenges to be addressed in enabling this capability. Firstly, the transfer of information from the MBSE specification to the MDAO environment needs to be identified and categorized. Organizations typically have static multi-fidelity workflows that may not be well suited for reconfiguration to optimize novel architectures or cater to the "architecting" process. Significant efforts in this area have led to developing MDAO frameworks such as [27] and [28] that feature reconfiguration capability and can support collaborative MDAO processes. However, an integration of MDAO methods to exclusively support system architecting activities is lacking. The lack of effective information exchange between system architecting frameworks and MDAO environments is a possible barrier against better integration, thereby leading to the question of how architecture description artifacts can be mapped to workflow elements in an MDAO specification.

Another aspect to consider is that of tool fidelity. It may prove feasible to obtain overall results at the conceptual design phase using a workflow constituting low fidelity tools. However, including high-fidelity analysis as part of the workflow on aircraft-level metrics would still need to be quantified in terms of uncertainty. Several approaches to formalizing the evaluation phase of system architecting have been proposed in the literature [29]–[31], and the integration of the architect's knowledge base has been demonstrated by [32], where system safety and performance analysis is shown in the context of system architecting.

Architecture definition explores the design space of architectural options which constitute combinations of specific technologies and functions [24]. The large set of system architecture combinations is checked for feasibility to derive a reduced set of architectures. Following this, the system architectures are visualized in a diagrammatic specification that could also involve more digital elements in the form of a model-based specification. The specified architecture is then evaluated for performance, and the one that best meets the user requirements is selected.

Finally, there are the aspects of incorporating safety and certification consideration which can be beneficial as the system model can hold this information. An MBSE and MDAO link would then allow this information to be added to the MDAO workflow in the form of constraints [33], [34]. However, attempting to run a large design space through the architecture evaluation process is inefficient and restrictive in terms of computational cost. Therefore, a means of prefiltering is required to establish a viable design space of safe and certifiable architectures, which leads back to the developments required in the architecture definition stage.

Overall, there is a disconnect between the fields of system architecting and MDAO, where system architecting is a highly iterative and manual process subject to reconfiguration, whereas MDAO is static and automated. A categorization of these disparities is presented in Tab. 1.

Tab. 1 Classification of the differences between MDAO and Systems Architecting

Category	Systems Architecting	MDAO
Development	Iterative	Mostly Static
Implementation	Manual/Some Automation	Mostly Automated
Fidelity	Mixed – Low/High	Mixed- Low/High
Formalization	Architect/System Engineer Knowledge	Established MDAO Frameworks

Recent developments have focused on providing more automation to MDAO implementation and execution [27], [35]. The MDAO formulation phase has been made more efficient through the use of centralized MDAO schema and MDAO definition environments that allow the consolidation of tool repositories as well as architecting and visualization of MDAO workflows [35]. These schemata ensure that an MDAO specification can be architected and executed in an automated manner. Overall, MDAO specification, implementation, and execution are mature and feature a high level of automation and potential for reconfiguration. However, the general trend of static MDAO workflows does remain in many organizations as their workflows are implemented in Process Integration and Design Optimization (PIDO) tools and require manual effort for reconfiguration.

In summary, the disconnect between the MDAO and system architecting is evident in disparities relating to development, fidelity, formalization, and reconfiguration. However, the need to integrate MDAO and system architecting is imperative, particularly for unconventional aircraft system configurations. A means of integrating MDAO and system architecting can be devised by identifying the information that needs to be shared between these two areas. An understanding of the current structure of MDAO related data can be garnered by analyzing recent MDAO exchange standards and implementation frameworks. This is described in the following section.

B. MDAO Process and Implementation Standards

The MDAO process is driven by the MDAO formulation phase, where the MDAO problem is addressed by consolidating a tool repository, selecting a solution strategy, and deriving an MDAO specification. The MDAO workflow is formulated in response to an MDAO problem, which dictates the associated disciplinary tools and MDAO solution strategy that is to be employed. Finally, a consolidation of MDAO tools and solution strategy is formalized in an MDAO specification containing information about tool interconnections, exchanged data, and runtime execution order. The MDAO formulation phase takes up a significant amount of time and is a crucial step in the success of any MDAO activity [36].

The MDAO specification needs to be implemented in a software environment and made executable, thus requiring a robust data framework within which information about MDAO workflow configuration can be stored, exchanged, and processed.

The Common MDAO Workflow Schema (CMDOWS) as defined by [35] along with the Common Parametric Aircraft Schema (CPACS) provides a means by which information pertaining to the MDAO workflow is captured within a neutral data format that can then be processed by software and visualization frameworks for execution and interaction [37]. The CMDOWS file stores three top-level information types relating to the MDAO specification: information, nodes, and connections. The information category contains data about the MDAO specification pertaining to problem definition, problem formulation. Nodes and constraints are described within the framework of a graph-based representation. Nodes contain executable blocks which further represent either analytical relations or design competencies. These offer sophisticated evaluations that cannot be represented by a single relation and can also be provided as a remote service. Connections hold information about the links between executable blocks and the order of execution of these blocks when following a particular solution strategy. A detailed description of the CMDOWS can be found in [35]. Overall, CMDOWS provides the required framework to transition from MDAO specification to execution within a software environment such as the Remote Component Environment (RCE) and enables collaboration through being amenable to interaction and visualization.

OpenMDAO is another MDAO framework within which MDAO problems are specified and executed [38]. Its object-oriented implementation uses classes specified as Components, Groups, Problems, and Drivers. The component class is used to implement relations, expressions, and even external code. Groups are containers for components, and drivers are used to implement a particular optimization strategy or DOE. Overall, it is seen that an MDAO specification requires the derivation of information from the MDAO problem and its storage in an exchangeable schema- either implicit as in OpenMDAO or explicit, as exemplified in the CMDOWS schema- for complete MDAO execution.

The extraction of data from the MDAO specification into the data schema and its subsequent execution is well handled by existing frameworks. However, the integration of an MDAO implementation into the system architecting process requires information derived from the architecture specification. Therefore, the specification needs to be detailed enough to include information about system physical components, the nature of exchanges, exchanged variables, and the level of detail included in the representation. However, this level of specification might be prohibitive for a large architectural design space, and therefore a reduced design space needs to be considered.

The established MDAO frameworks allow an MDAO workflow to be specified and executed in software frameworks. Once an MDAO architecture is established, it can be used to complete numerous computations. Moreover, the benefits of a single point MDA can also be exploited once the workflow is set up. However, reconfiguring the workflow is a major challenge, and recent advances have gone a long way in addressing this issue [35]. Extensive automation of workflow implementation and experience with multi-fidelity MDAO in an industrial context for the development of a complex product shows that MDA and MDAO results are reliable [1]. On the other hand, systems architecting is a relatively manual and iterative process, requiring knowledge from the architect.

Moreover, systems architectures are more prone to reconfiguration, and the impact of architectural choices needs to be quantified quickly to aid the decision-making process. An architectural layer exists where system architecture choices are evaluated for key metrics and reconfigured according to the architect's experience [39]. This type of flexibility may not be easily implementable in an MDAO framework.

IV. Proposed Framework Concept

The proposed framework aims to link MBSE and MDAO to facilitate the efficient architecting of aircraft systems. Fig. 2 describes the MDAO process in the context of systems architecting. The system architecture design space is represented using a graph-based descriptor and filtered for feasibility based on predefined rules. This results in a reduced set of architectures that are modeled within an MBSE environment. Here, MBSE is introduced as a link between MDAO and systems architecting but also as an integral part of the systems architecting process. The MDAO environment consists of the MDAO specification containing information about the tools, interconnections, and choices about MDAO architecture, as well as the designation of different design variables, objectives, and constraints. The MDAO data schema holds this information in a structured manner which then allows MDAO execution through a PIDO platform, as discussed in Section III A. Furthermore, it is envisioned that the model-based specification aids the development process by reusing model artifacts beyond the conceptual design phase. It is also noted that ensuring an architecture specification early in the design process helps establish a traceable baseline for later reference.

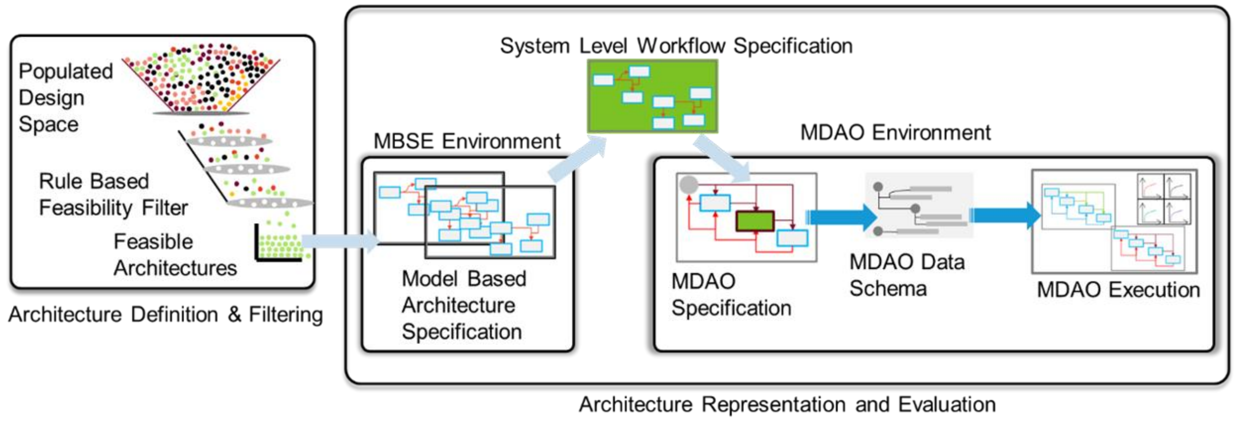

Populated Design Space
Rule Based Feasibility Filter
Feasible Architectures
Architecture Definition & Filtering
System Level Workflow Specification
MBSE Environment
Model Based Architecture Specification
MDAO Environment
MDAO Specification
MDAO Data Schema
MDAO Execution
Architecture Representation and Evaluation

Fig. 2 Proposed framework for integrating MBSE and MDAO for systems architecting

The architecture specification uses a catalog of architectural artifacts, which include logical and physical and components that have been defined with a granularity level that is appropriate for use in conceptual design. The benefit of having the artifacts defined early in the process is that once a system architecture has been chosen, the basic artifacts already contained in the model-based specification can be easily enriched and developed further as the design process continues. These modeling artifacts are mapped to specific disciplinary tools by categorizing the inputs and outputs that each modeling artifact is assigned.

As an example, for the enhanced use of the MBSE specification, the system architectures can now be used for safety assessment, e.g., performing a functional hazard analysis or the extraction of fault trees. The authors will explore these enhanced capabilities in future work; the proposed framework prepares the integration of this capability.

The remaining few architectures can now be evaluated in an overall integrated, aircraft level and system level MDAO workflow. The information for the MDAO specification is extracted from the architecture specification model. The architecture specification helps configure the subsystem-level workflows in conjunction with information from the top-level requirements such as range, payload, and mission profile. These subsystem workflows are then formally specified and integrated into an MDAO specification. The aircraft level workflow is assumed to have already been created based on the available tool repository.

The resulting specification is incorporated into an existing MDAO specification data schema such as CMDOWS. This allows the MDAO workflow to be executed within an existing MDAO implementation framework such as RCE and OPTIMUS [40][41]. A single point iteration is conducted, and the results of the MDA for each architecture are compared. At this point, architectures with unacceptable performance characteristics may be discarded. The metrics of evaluation are those present at the aircraft level, such as MTOW, fuel burn, and operating cost. Each architecture is optimized accordingly, and the results are compiled to compare their relative performance. A multi-attribute decision-making approach is employed to make the final selection of system architecture from the published MDAO results.

The presented framework employs a model-based architecture specification to enrich the information being supplied to the MDAO specification. It uses two loops of evaluation to utilize the capabilities of MDA and MDAO, respectively. The following case study illustrates the envisioned information to be exchanged between the MBSE specification and the MDAO environment to enhance the system architecting process.

V. Case Study: Modelling and extraction of properties from a sample system architecture in Capella

Within the scope of this study, specific model elements are enriched by adding information to quantify characteristic properties that support system-level sizing tools such as actuator technology, operating pressure, and actuator to surface assignment for a primary flight control system architecture. A means of extracting this information from the underlying data model is presented in an automatization algorithm. The extracted properties are then relayed to associated tools, and the results are written back to the Capella model elements in the form of additional model properties. An initial approach to integration with AGILE 4.0 project workflows is also described where-in a static workflow is considered, and tool inputs are extracted from Capella model properties and provided to the workflow specification data model.

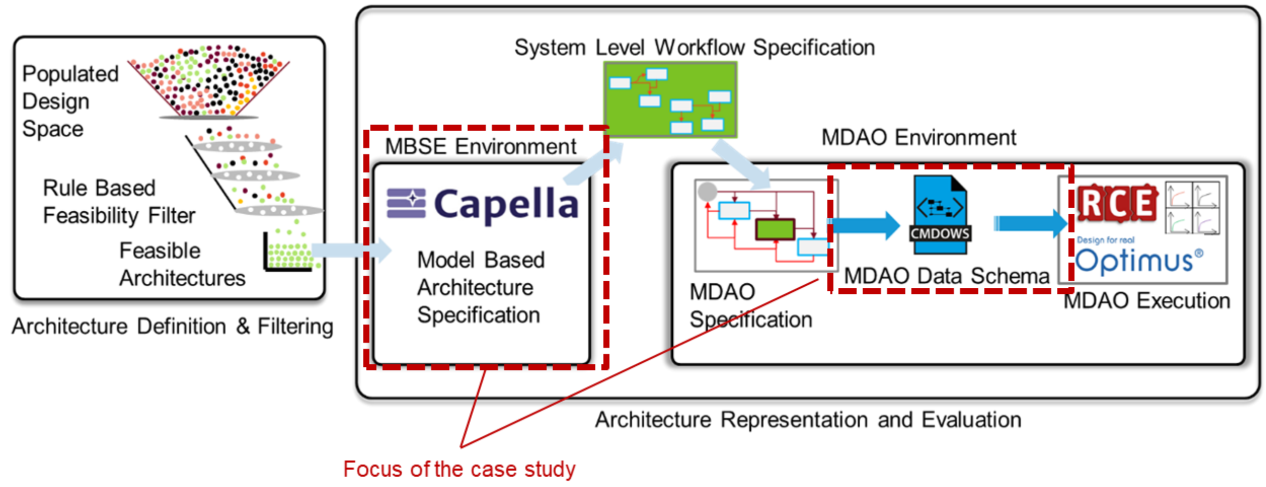


Fig. 3 Tool selections for MBSE-MDAO integration framework

Fig. 3 shows the specific tools implemented to explore the integration between MBSE and MDAO. Here, the architecture specification is built in Capella, then enriched with specific input parameters corresponding to tools in the system-level workflow, which is translated into a graph-based MDAO specification. The reference of each input parameter in the CPACS file is stored in the CMDOWS schema along with information on tool interconnections and tool execution order. The CMDOWS file is then processed by a PIDO framework such as RCE, where the workflow is executed and results are obtained.

This case study will explore the following aspects of linking the architecture specification Capella with a system-level MDAO workflow, using an example of the flight control system and tools developed in the frame of the AGILE4.0 project:

1. Enriching the specification model with system-level tool input parameters
2. Extraction of parameters from MBSE (Capella) data model
3. Integration of extracted parameters into an MDAO workflow

The Capella tool is used for the MBSE portion. In Capella, four different levels of abstraction (the operational level, system level, logical level, and physical level) are available to specify a system. Here, the physical level is selected to establish the interface to the MDAO workflow. The link could potentially be established at other specification levels; however, this consideration is out of scope for this paper and will be discussed in future work.

Fig. 4 shows the main physical components, the logical components, and the associated functions of a portion of the flight control system architecture. Functional exchanges are depicted, and the physical links between components are also specified. Here, the key components are the hydraulic actuators, control surfaces, and elements representing the supply of secondary power to each actuator. The flows of signal and power are highlighted using the "functional chain" feature in Capella, wherein individual functional exchanges can be selected and converted into a continuous chain. In this model, all the actuators are of the Electrohydraulic Servo Actuator (EHSA) type, of which two are assigned to each elevator control surface. The type of actuator and the allocation per control surface are important for the sizing of actuators. Each of these elements has specific properties that are used by the system level sizing and performance workflows to estimate mass, power demand, and other metrics using empirical or physics-based methods.

However, the extraction of information is challenging as most of this information (such as number and type of actuator) can be inferred from the diagram and other information, which can include hydraulic pressure, supplied power, and component efficiencies, have to be extracted from the underlying data model. This aspect will be covered in the following sections.

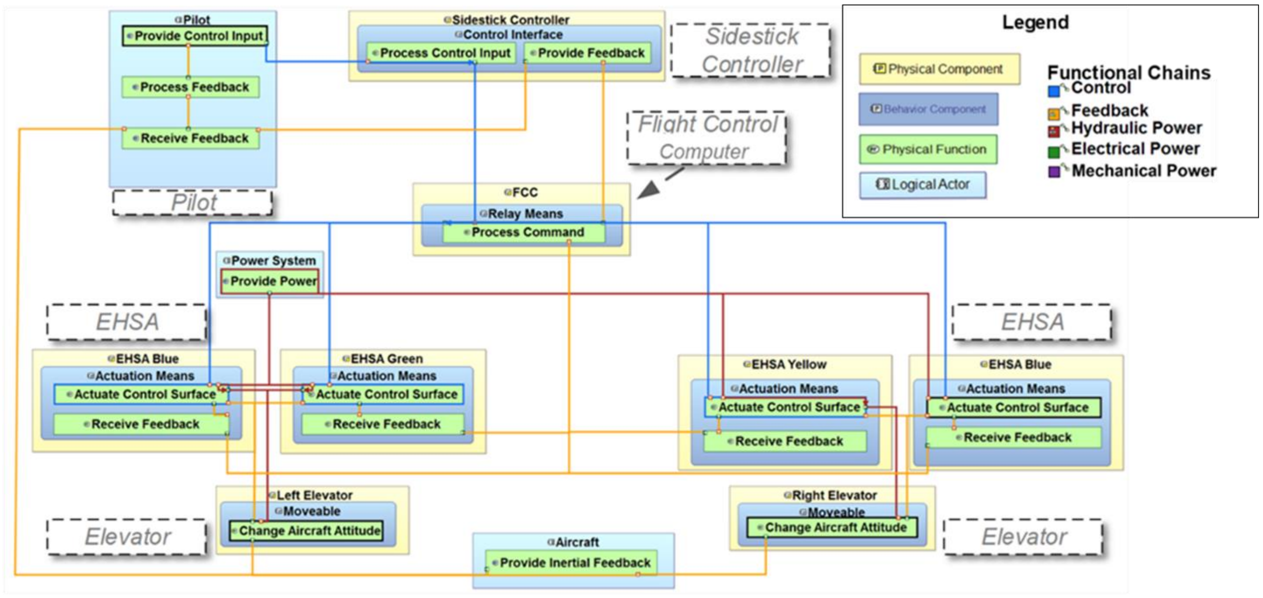
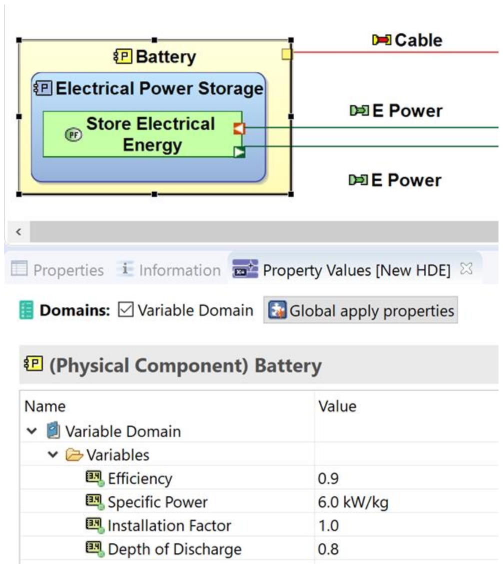


Fig. 4: Physical architecture model in Capella for a portion of a flight control system (pitch control system), from [20] **(EHSA stands for Electrohydraulic Servo Actuator)**

An architecture specification developed in Capella already supports additional information such as component descriptions, architecture summaries, model developmental status, and model validation information. However, Capella does not natively support the addition of custom or model-specific properties and the interrogation of model elements to access these properties. However, an open-source plugin for Capella called the Property Value Management Tool (PVMT) [42] which allows the definition of properties of different data types (string, float, integer, and others), as well as the implementation of specific rules or conditions with which the properties are applied.

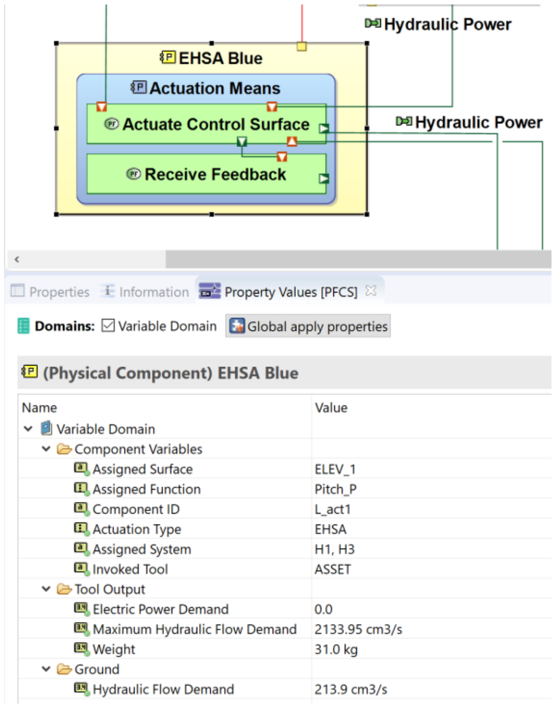
Furthermore, properties can be made applicable to specific modeling levels such as the system or physical level. They can also be set to specific modeling artifacts such as exchanges or functions.

As an example, Fig. 5 (a) shows properties such as efficiency, specific power, installation factor, and depth of discharge applied to a physical component representing a battery. These properties can be used to provide input to a battery sizing tool that will provide the battery's overall weight. Fig. 5 (b) shows properties that are inputs to an actuator sizing tool, such as actuation type and assigned surface. Tool outputs such as the weight and flow demand can be categorized separately and specified within the Capella workbench. Properties can also be directly accessed from the underlying data model in Capella. This aspect is explored in the following section.



(Physical Component) Battery

Name	Value
Variable Domain	
Variables	
Efficiency	0.9
Specific Power	6.0 kW/kg
Installation Factor	1.0
Depth of Discharge	0.8



(Physical Component) EHSA Blue

Name	Value
Variable Domain	
Component Variables	
Assigned Surface	ELEV_1
Assigned Function	Pitch_P
Component ID	L_act1
Actuation Type	EHSA
Assigned System	H1, H3
Invoked Tool	ASSET
Tool Output	
Electric Power Demand	0.0
Maximum Hydraulic Flow Demand	2133.95 cm3/s
Weight	31.0 kg
Ground	
Hydraulic Flow Demand	213.9 cm3/s

(a) Component characteristics for a battery **(b) Component characteristics and sizing results for a flight control actuator**

Fig. 5: Allocation of properties to battery physical component in Capella using the PVMT add-on

A. Extraction of Parameters from the Capella Data Model

Capella is built on the Eclipse Modelling Framework and thus supports customization through modifying the core functionality and the development of add-ons [43]. The underlying framework consists of a structured data model which stores all information. The Capella model can be interrogated in the following ways:

1. Visually using the workbench and diagram editors[5]
2. Accessing the underlying data model and schema

Visual inspection of the Capella model is achieved by the system architect manually interacting with the diagram editor and accessing the PVMT viewpoint tab. The system architect then will have to transcribe the information manually to either an excel sheet, intermediate data format, or directly to the input file of a system-level workflow or tool.

Reading and writing to and from the underlying data model is a practical solution to integrating an MBSE specification in Capella with external tools. This data model is an XML schema that contains all the information about the system architecture specification, including the names of functions, logical components, physical components, functional exchanges, component exchanges, physical links, owned components (functions, logical and physical components), as well as information on assigned properties and their values. Furthermore, the location of graphical elements and changes made to the model are also stored within this schema.

[5] Querying the model from within the workbench user interface is also considered here.

The data model consists of two files to which all the information is written every time a user saves changes to the model. These are the ".melodymodeller" – which is now called ".capella" as of version 5.0.0 and the ".aird" files. The ".aird" file contains information on active diagrams within a Capella project, including the location and formatting of elements in a diagram and a record of changes made to the model. The ". capella" file stores information about each model element, including characteristics, assignments, links, input and output ports, as well as any assigned property values

Each model artifact is assigned a unique alphanumeric sequence as an identifier within the schema, and links between elements such as functional chains and physical links reference these identifiers in specifying the source and target of each exchange. Although the schema is well structured, the size and tag names can make navigation cumbersome. In order to simplify the traversal of different tags for property extraction and the automation of the process, a simplified schema is created which contains only the tags that are accessed to extract model element properties. The mapping from the actual schema tags to a simplified version is shown in Tab. 2, supplemented by an illustration in the Appendix.

Tab. 2: Mapping between Capella schema and simplified schema tag names

Capella Schema Tag Name	Simplified Schema Tag Name
Root: org.polarsys.capella.core.data.capellamodeller:Project	Project
Root:ownedPropertyValuePkgs	Property Value Packages
Child:owned PropertyValueGroups	Property Value Groups
Child:ownedPropertyValues	Property Values
Root: ownedModelRoots	Models
Child: ownedArchitectures	Architectures
Child:ownedPhysicalComponentPackage	Physical Architecture
Child:ownedPhysicalComponents	Components
Child:ownedPropertyValueGroups	Property Value Groups
Child:ownedPropertyValues	Property Values

An algorithm for extracting model properties from the ". Capella" file is explained below in Fig. 6. This process is based on the following assumptions:

1. A predefined list of components is created in Capella, each with its name and unique identifier.
2. A list of properties is defined with the PVMT tool, and variable names are included in the "key" field within the PVMT editor.

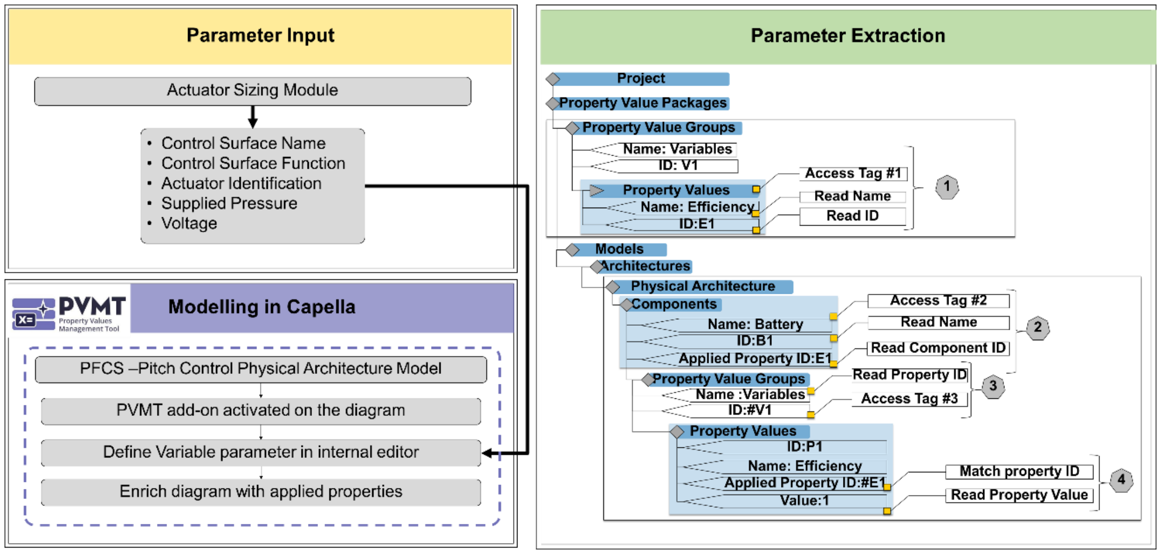


Fig. 6: Process for parameter input and extraction in Capella

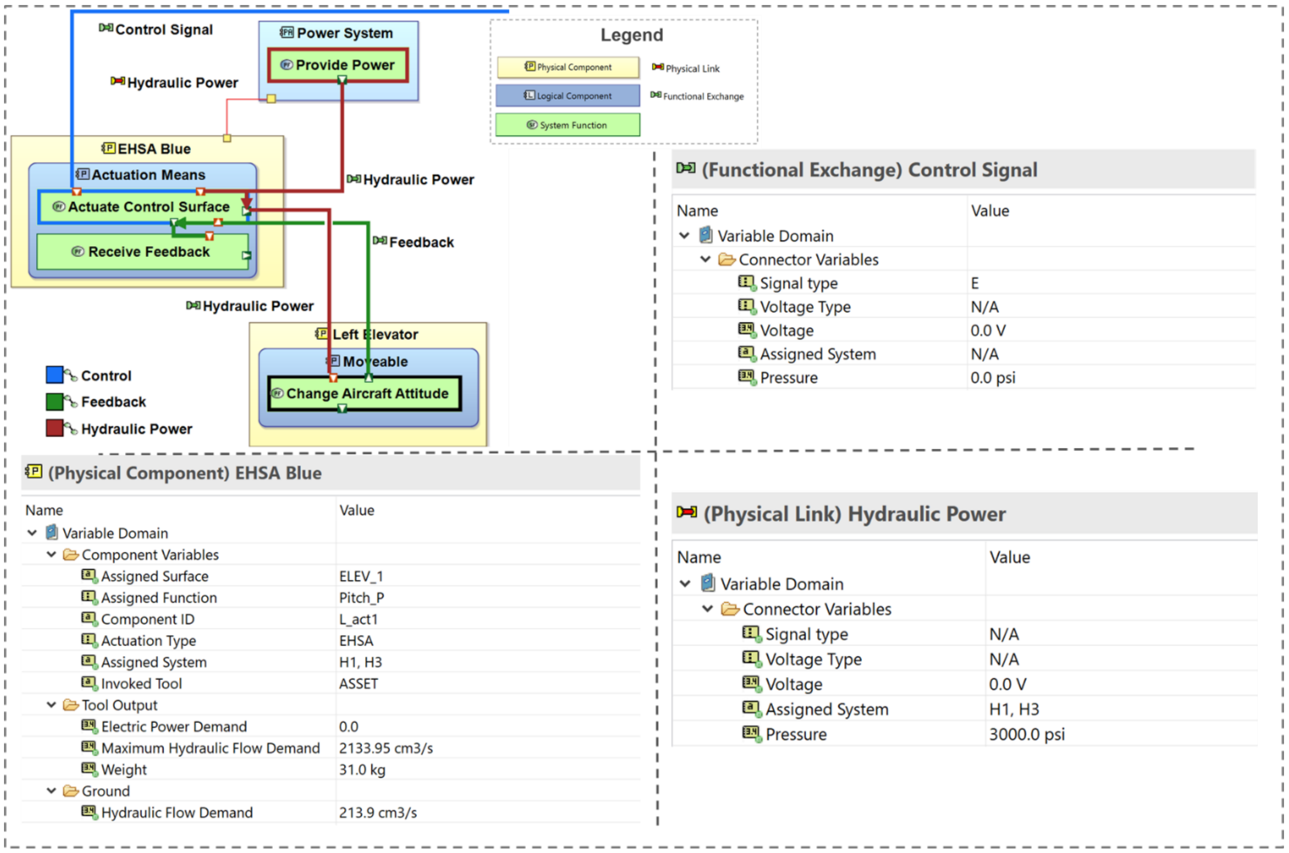
First, the inputs are compiled and added to the Capella model through the PVMT tool. To do so, one needs to activate the PVMT viewpoint in Capella and define and name each property's data type. Once the properties are defined, values are assigned to each field, and the model must be saved. One can extract parameters by first reading the top-level property value group tag and then reading the name and ID of the child property value tag. Secondly, one can access the components tag and parse for the name, ID, and applied property ID of each physical component. The child property value groups tag is then accessed, and the ID value is read. Finally, the property values tag, which itself is a child of the property value groups tag assigned to the physical component, -is accessed. Here a match is made between the applied property ID and the property value ID defined in step 1 of Fig. 6, and the value of the property is stored.

In some cases, when the number of inputs from system architecture to a system-level tool or workflow is relatively high, it may be cumbersome to complete an exhaustive specification of properties in Capella. In such cases, parsing the Capella schema in the manner described above can be aided by adding inferential logic from the system architecture specification to define the additional tool input values. This functionality is demonstrated for system architecture shown in Fig. 5, when linking it to an actuator sizing tool that is part of a system-level workflow. The inputs and outputs of this tool are defined in Tab. 3.

Tab. 3: Tool inputs derived from different Capella elements

Required Tool Inputs	Source of Value	Referenced Capella Element
Control Surface name	Specified	Physical Component Property
Control Surface function	Specified	Physical Component Property
Pressure	Specified	Physical Link Property
Voltage	Specified	Physical Component/Link Property
Voltage type	Specified	Physical Component/Link Property
Technology	Specified/Inferred	Physical Component Property
Flight critical system designation	Specified	Physical Component Property
Association to System	Inferred	Physical Component Property
No. of Hydraulic Actuators	Inferred	Physical Component Property
No. of Electric Actuators	Inferred	Physical Component Property
Surface to Actuator Assignment	Inferred	Physical Component Property

In Fig. 7, the physical links from the power system element to the actuator elements are shown to have assigned properties that include power type and pressure. When reading these parameters, one can deduce that the actuator connected to this power system type uses hydraulic power. Similarly, one can assign a parameter called 'signal type' to the input signal that is represented by a functional exchange. In this case, the system architect can deduce that the actuator is electrically controlled as the control signal is of type 'electric.' Combining the logic of these two inferences allows for the actuator to be hydraulically powered and electrically controlled. This results in the identification of the actuator type, and iterating through all the actuator components allows the total number of actuators of a given type to be determined.



(Functional Exchange) Control Signal

Name	Value
Variable Domain	
Connector Variables	
Signal type	E
Voltage Type	N/A
Voltage	0.0 V
Assigned System	N/A
Pressure	0.0 psi

(Physical Component) EHSA Blue

Name	Value
Variable Domain	
Component Variables	
Assigned Surface	ELEV_1
Assigned Function	Pitch_P
Component ID	L_act1
Actuation Type	EHSA
Assigned System	H1, H3
Invoked Tool	ASSET
Tool Output	
Electric Power Demand	0.0
Maximum Hydraulic Flow Demand	2133.95 cm3/s
Weight	31.0 kg
Ground	
Hydraulic Flow Demand	213.9 cm3/s

(Physical Link) Hydraulic Power

Name	Value
Variable Domain	
Connector Variables	
Signal type	N/A
Voltage Type	N/A
Voltage	0.0 V
Assigned System	H1, H3
Pressure	3000.0 psi

Fig. 7: Properties applied to physical components and physical links as a basis for inferential logic

Similarly, the assignment of actuators to control surfaces may be determined by iterating through all physical links. Figure 9 shows an example of an actuator and control surface combination. Here, the physical links represent the hydraulic power that is supplied to the actuator. Inspection of the properties contained within these links reveals additional details such as pressure and assigned hydraulic system identifiers.

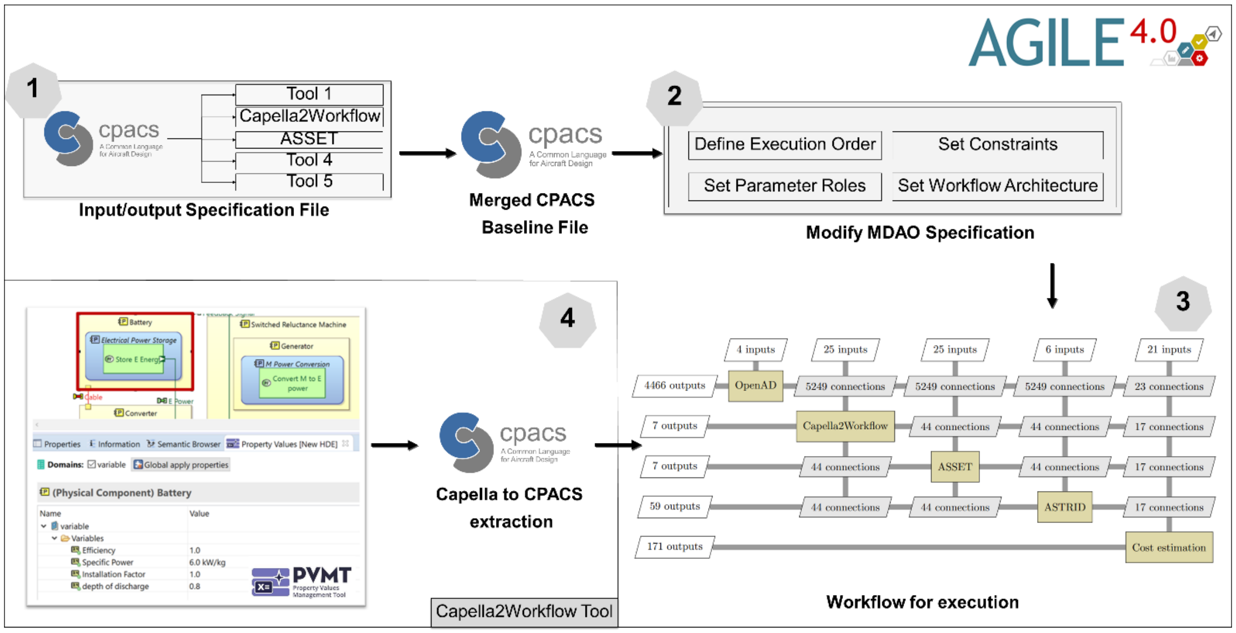
C. Methodology to transfer parameters into AGILE 4.0 workflows

For the example of the AGILE4.0 project, the MDAO workflow specification is created collaboratively using the KADMOS tool and the process platform KE-chain [44], [45]. In KE-chain, each tool owner specifies a so-called design competence. For each of these design competencies, the input and output parameters are initialized using separate CPACS files as an interface standard. Additional information, such as tool parameters, versioning, and ownership information, may also be provided at this stage. Following this stage, a merged CPACS baseline file is created by linking all the individual input and output files. This merged baseline carries the information between tools during workflow execution. The next steps involve modifying the workflow architecture, specifying the tool execution order, setting up design parameters, and specifying constraints, objectives, and state variables. A CMDOWS file is generated, which can then be used to execute the workflow in RCE along with the input baseline CPACS file.

Transferring parameters into the MDAO workflow can be achieved by the following means:

1. directly modifying the CPACS baseline file and the CMDOWS file
2. packaging extracted parameters and integrating them with the AGILE environment to generate a CMDOWS file
3. integrating the extracted parameters in the form of an example tool

The third approach is presented here as it is the most straightforward. However, its limitations include the need always to have an existing baseline workflow. Furthermore, only the system-level tool inputs, i.e., the inputs that do not change and are not overwritten upon workflow execution, are considered. Fig. 8 shows the overall process and the resulting workflow with the integration of parameters derived from Capella.



Name	Value
variable	
Variables	
Efficiency	1.0
Specific Power	6.0 kW/kg
Installation Factor	1.0
depth of discharge	0.8

Fig. 8: Process to integrate Capella with AGILE4.0 workflow

The system architecture from Figure 5 is used as input to the ASSET[6] tool, which performs the actuator sizing and returns the overall actuator weight and power demand. This tool derives its input from the parameters specified in Capella. The functionality of parameter extraction is integrated within the AGILE workflow as the Capella2Workflow tool. This is performed by defining the input and output within the AGILE workbench. Capella2Workflow provides the parameter values directly to ASSET, which further passes its output to the ASTRID[7] tool [46], which performs the sizing of all other systems. The parameters extracted from Capella are written directly to the output CPACS file of Capella2Workflow, which then serves as the input file for the ASSET tool. The workflow output is written to the baseline CPACS file. The relevant parameters are extracted and written back to the Capella data model schema, which is then available for inspection within the Capella workbench.

D. Discussion

Overall, the presented case study demonstrates how properties can be added to Capella model elements and how they can be extracted by parsing the data model schema. The addition of properties is flexible with the ability to define multiple property groups. In this study, property extraction is performed by manually parsing the Capella data model. However, the parsing approach is generic and can be performed manually or automated with the appropriate tools. The challenge of automating this process is reduced by understanding and simplifying Capella's data schema structure.

An initial approach to integrating Capella, as an MBSE tool, with the system-level MDAO tools in the AGILE4.0 workflow is also presented. The current approach is straightforward as it packages the extraction of Capella model properties as a tool that provides these properties as an input to other tools in the workflow. The main limitation of this approach is that the system-level workflow, including the selected tools and execution order, needs to be predefined. Although the model properties include a field for the name of the tool to which the parameters will be passed, the ability to reconfigure the workflow based on the referenced tool is not yet supported. The use of different modeling elements to infer tool inputs that are not explicitly specified is also shown. However, an algorithm to make such inferences is required alongside automation to make this process efficient.

Overall, the presented study explores the practical aspects of linking system architectures specified in the Capella MBSE tool to system-level tools and the AGILE 4.0 MDAO workflow. As a result, areas of improvement to mature the MBSE-MDAO link functionality have been identified.

[6] ASSET: Aircraft System Sizing Estimation Tool in development at Concordia University
[7] ASTRID: Aircraft system sizing and performance estimation tool developed by the Politecnico di Torino

VI. Conclusion and future work

The development of novel aircraft featuring unconventional and more electric systems architectures drives the need to incorporate systems architecting as a critical activity in conceptual design. In addition to constructing integrated system sizing and performance analysis frameworks, the optimization of system architectures must also be considered by implementing MDAO workflows.

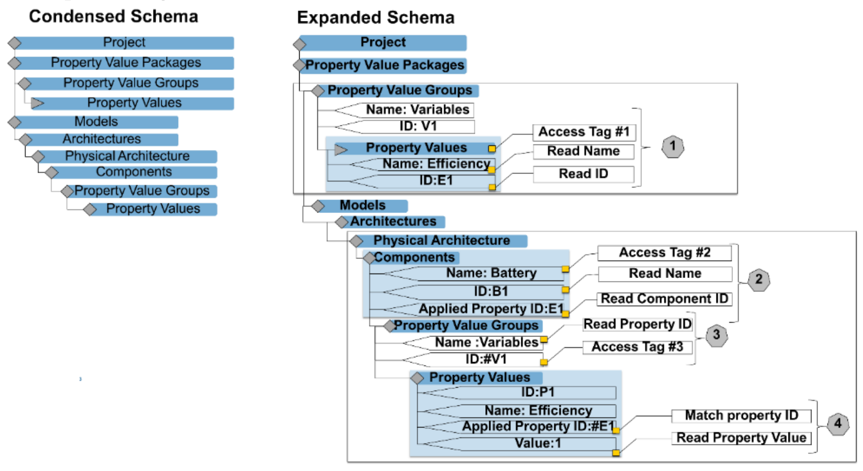
This paper highlights the iterative nature of systems architecting and the recent advances that have been made in the specification and automation of MDAO workflows. Furthermore, the gap in the use of MBSE for systems architecting in conceptual design is highlighted. Herein, a framework has been proposed that treats the following: a means of adding tool input properties to elements within the Capella MBSE tool, a means of extracting the applied properties to supply system-level tools, and an initial concept of integrating with established MDAO specification and visualization environment developed by the AGILE 4.0 project. A demonstration of the application of properties in Capella using the PVMT plugin as well as their extraction by parsing the Capella data model has been made through a case study of a pitch control system architecture.

Future development of this framework will include the improvement of automation and the ability to reconfigure the workflow based on the system architecture specification. Additionally, the aspects of safety and certification described briefly in this paper will be presented in greater detail. Also, the automated analysis of a large design space and the exploration of required modeling levels in MBSE will be investigated.

Overall, this paper presents a practical contribution to linking MBSE and MDAO and paves the way for better integration of MBSE into the aircraft design process, allowing the use of MBSE from conceptual design onwards and enabling more detailed system analysis, such as safety analysis starting in conceptual design, based on architecture models.

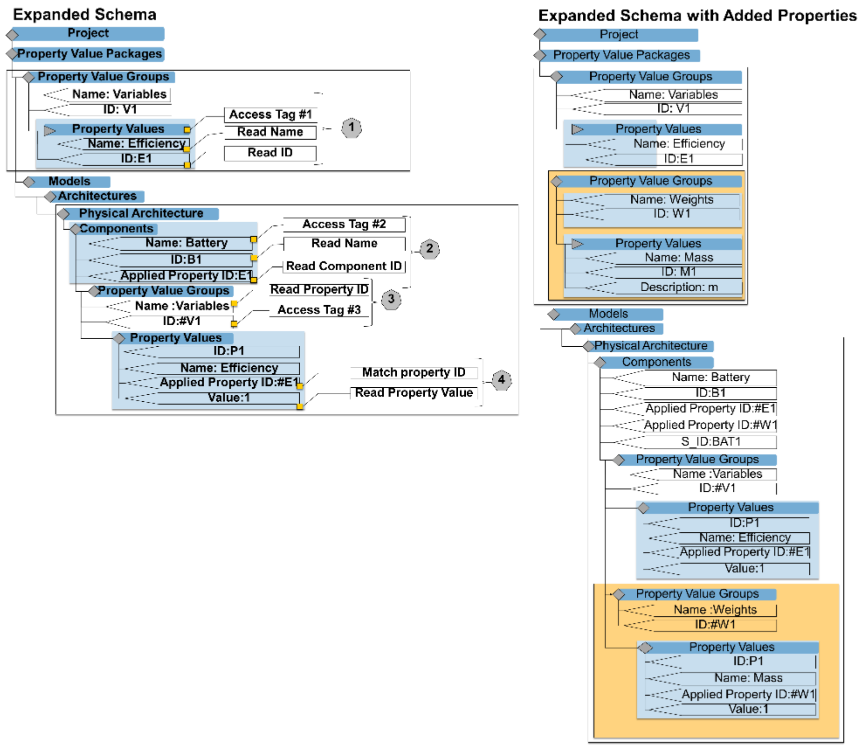
VII. Appendix

Fig.A1 shows the simplified Capella data schema with the tag names that represent different types of model information. An expanded version is also included with annotations describing the steps required to extract property information from the schema. These steps are labeled from 1 to 4 and include sub-steps about accessing and opening specific elements of each parent tag.



A 1: Condensed and expanded versions of the simplified Capella XML Schema

Fig. A2 shows the expanded schema and an additional section that is included when a user manually modifies the schema to add a property value. These additions are shown to be both within the Property Value Packages tag and within the Physical Architecture parent tag.



A 2: Capella XML data schema: the highlighted section shows where properties have been added manually

Acknowledgments

We acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC), Grant Number CRDPJ 538897-19 and RGPIN/5515-2019, and the Consortium de recherche et d'innovation en aérospatiale au Québec (CRIAQ). The presented work also benefited from the authors' collaboration within the AGILE4.0 project, which receives funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 815122.

The authors thank Bastiaan Beijer at KE-works for his help and feedback for implementing the case study within the AGILE4.0 context.

References

[1] P. Piperni, A. DeBlois, and R. Henderson, "Development of a Multilevel Multidisciplinary-Optimization Capability for an Industrial Environment," AIAA J., vol. 51, no. 10, pp. 2335–2352, 2013.

[2] J. Gray, K. T. Moorey, and B. A. Naylorz, "OpenMDAO: An open source framework for multidisciplinary analysis and optimization," in 13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference 2010, 2010, doi: 10.2514/6.2010-9101.

[3] P. D. Ciampa and B. Nagel, "The AGILE paradigm: The next generation of collaborative MDO," in 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2017, doi: 10.2514/6.2017-4137.

[4] "AGILE 4.0 – Towards cyber-physical collaborative aircraft development." https://www.agile4.eu/ (accessed Jun. 18, 2021).

[5] P. S. Prakasha et al., "Collaborative system of systems multidisciplinary design optimization for civil aircraft: AGILE EU project," in 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2017, doi: 10.2514/6.2017-4142.

[6] M. Fioriti, L. Boggero, F. Tomasella, A. Mirzoyan, A. Isyanov, and P. S. Prakasha, "Propulsion and on-board system integration for regional, medium and long range jet with different level of systems electrification in the agile project," in Joint Propulsion Conference, 2018, doi: 10.2514/6.2018-4847.

[7] L. Boggero, M. Fioriti, S. Corpino, and P. D. Ciampa, "On-board systems preliminary sizing in an overall aircraft design environment," in 17th AIAA Aviation Technology, Integration, and Operations Conference, 2017, doi: 10.2514/6.2017-3065.

[8] E. Caliò, F. Di Giorgio, and M. Pasquinelli, "Deploying Model-Based Systems Engineering in Thales Alenia Space Italia," in CIISE, 2016, Accessed: Mar. 20, 2019. [Online]. Available: https://www.polarsys.org/capella.

[9] X. Hai, S. Zhang, and X. Xu, "Civil aircraft landing gear brake system development and evaluation using model based system engineering," in 36th Chinese Control Conference (CCC), 2017, pp. 10192–10197, doi: 10.23919/ChiCC.2017.8028981.

[10] T. J. Bayer et al., "Model Based Systems Engineering on the Europa Mission Concept Study," 2012. Accessed: Mar. 20, 2019. [Online]. Available: https://trs.jpl.nasa.gov/bitstream/handle/2014/45000/11-5594_A1b.pdf?sequence=1.

[11] B. Malins, "Model-Based Systems Engineering Applied to a Hospital Department," in INCOSE International Workshop, 2016.

[12] P. D. Ciampa, B. Nagel, and G. La Rocca, "A MBSE Approach to MDAO Systems for the Development of Complex Products," in AIAA Aviation Forum, 2020, doi: 10.2514/6.2020-3150.

[13] IEEE, "ANSI/IEEE 1471-2000: IEEE Recommended Practice for Architectural Description of Software-Intensive Systems," 2000. Accessed: May 17, 2021. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=875998.

[14] Air Transport Association, "ATA 100." http://www.s-techent.com/ATA100.htm (accessed Mar. 12, 2019).

[15] S. Liscouët-Hanke, "A model-based methodology for integrated preliminary sizing and analysis of aircraft power system architectures,"PhD Thesis, Institut National des Sciences Appliquées de Toulouse, 2008.

[16] I. Chakraborty and D. N. Mavris, "Heuristic Definition, Evaluation, and Impact Decomposition of Aircraft Subsystem Architectures," in 16th AIAA Aviation Technology, Integration, and Operations Conference, Jun. 2016, doi: 10.2514/6.2016-3144.

[17] SAE International, "ARP4754A: Development of Civil Aircraft and Systems," 2011. doi: 10.1002/jmr.585.

[18] F. Mhenni, J.-Y. Choley, N. Nguyen, and C. Frazza, "Flight Control System Modeling with SysML to Support Validation, Qualification and Certification," in IFAC-PapersOnLine, Dec. 2016, vol. 49, pp. 453–458, doi: 10.1016/j.ifacol.2016.07.076.

[19] S. Liscouet-Hanke, B. R. Mohan, P. Jeyarajan Nelson, C. Lavoie, and S. Dufresne, "Evaluating a Model-Based Systems Engineering approach for the conceptual design of advanced aircraft high-lift system architectures," in Canadian Aeronautics and Space Institute AERO 2017, May 2017.

[20] S. Liscouët-Hanke and A. Jeyaraj, "A Model-Based Systems Engineering approach for efficient flight control system architecture variants modelling in conceptual design.," in International Conference on Recent Advances in Aerospace Actuation Systems and Components, 2018, pp. 34–41.

[21] I. Chakraborty and D. N. Mavris, "Heuristic Definition, Evaluation, and Impact Decomposition of Aircraft Subsystem Architectures," in 16th AIAA Aviation Technology, Integration, and Operations Conference, Jun. 2016, doi: 10.2514/6.2016-3144.

[22] S. Jimeno, A. Riaz, M. D. Guenov, and A. Molina-Cristobal, "Enabling interactive safety and performance trade-offs in early airframe systems design," in AIAA Scitech Forum, Jan. 2020, vol. 1 PartF, pp. 1–16, doi: 10.2514/6.2020-0550.

[23] I. Chakraborty and D. N. Mavris, "Integrated Assessment of Aircraft and Novel Subsystems Architectures in Early Design," J. Aircr., vol. 54, no. 4, pp. 1268–1282, 2017, doi: 10.2514/1.c033976.

[24] J. H. Bussemaker, P. D. Ciampa, and B. Nagel, "System architecture design space exploration: An approach to modeling and optimization," in AIAA Aviation Forum, 2020, vol. 1 PartF, pp. 1–22, doi: 10.2514/6.2020-3172.

[25] S. Liscouët-Hanke, J. C. Maré, and S. Pufe, "Simulation framework for aircraft power system architecting," J. Aircr., vol. 46, no. 4, pp. 1375–1380, 2009, doi: 10.2514/1.41304.

[26] M. Fioriti, L. Boggero, F. Tomasella, A. Mirzoyan, A. Isyanov, and P. S. Prakasha, "Propulsion and on-board system integration for regional, medium and long range jet with different level of systems electrification in the agile project," in Joint Propulsion Conference, 2018, doi: 10.2514/6.2018-4847.

[27] J. Gray, K. T. Moorey, and B. A. Naylorz, "OpenMDAO: An open source framework for multidisciplinary analysis and optimization," in 13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2010, doi: 10.2514/6.2010-9101.

[28] P. D. Ciampa and B. Nagel, "The AGILE paradigm: The next generation of collaborative MDO," in 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2017, doi: 10.2514/6.2017-4137.

[29] I. Chakraborty and D. N. Mavris, "Assessing Impact of Epistemic and Technological Uncertainty on Aircraft Subsystem Architectures," in 16th AIAA Aviation Technology, Integration, and Operations Conference, Jun. 2016, doi: 10.2514/6.2016-3145.

[30] M. D. Guenov, A. Riaz, Y. H. Bile, A. Molina-Cristobal, and A. S. J. van Heerden, "Computational framework for interactive architecting of complex systems," Syst. Eng., Feb. 2020, doi: 10.1002/sys.21531.

[31] T. Lammering, "Integration of aircraft systems into conceptual design synthesis, "PhD Thesis, RTWH Aachen, Aachen, 2014.

[32] S. Jimeno, A. Molina-Cristobal, A. Riaz, and M. Guenov, "Incorporating Safety in Early (Airframe) Systems Design and Assessment," in AIAA Scitech Forum, Jan. 2019, doi: 10.2514/6.2019-0553.

[33] P. Schmollgruber et al., "Use of a Certification Constraints Module for Aircraft Design Activities," in 17th AIAA Aviation Technology, Integration, and Operations Conference, Jun. 2017, doi: 10.2514/6.2017-3762.

[34] M.-H. Bleu-Laine, M. V. Bendarkar, J. Xie, S. I. Briceno, and D. N. Mavris, "A Model-Based System Engineering Approach to Normal Category Airplane Airworthiness Certification," in AIAA Aviation Forum, Jun. 2019, doi:

10.2514/6.2019-3344.

[35] I. van Gent, G. La Rocca, and M. F. M. Hoogreef, “CMDOWS: a proposed new standard to store and exchange MDO systems,” CEAS Aeronaut. J., vol. 9, no. 4, pp. 607–627, 2018, doi: 10.1007/s13272-018-0307-2.

[36] I. van Gent, G. La Rocca, and L. L. Veldhuis, “Composing MDAO symphonies: graph-based generation and manipulation of large multidisciplinary systems,” in 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, American Institute of Aeronautics and Astronautics, 2017.

[37] 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, 2020.

[38] J. S. Gray, J. T. Hwang, J. R. R. A. Martins, K. T. Moore, and B. A. Naylor, “OpenMDAO: an open-source framework for multidisciplinary design, analysis, and optimization,” Struct. Multidiscip. Optim., vol. 59, no. 4, pp. 1075–1104, Apr. 2019, doi: 10.1007/s00158-019-02211-z.

[39] M. D. Guenov, A. Riaz, Y. H. Bile, A. Molina-Cristobal, and A. S. J. van Heerden, “Computational framework for interactive architecting of complex systems,” Syst. Eng., Feb. 2020, doi: 10.1002/sys.21531.

[40] D. Seider, M. Litz, A. Schreiber, P. M. Fischer, and A. Gerndt, “Open source software framework for applications in aeronautics and space,” in IEEE Aerospace Conference Proceedings, 2012, doi: 10.1109/AERO.2012.6187340.

[41] “Engineering Process Integration | Noesis Solutions | Noesis Solutions.” https://www.noesissolutions.com/our-products/optimus/engineering-process-integration (accessed Jun. 17, 2021).

[42] J. Navas, “Easily enrich capella models with your own domain extensions,” 2018. https://www.slideshare.net/Obeo_corp/webinar-november-22th-easily-enrich-capella-models-with-pvmt (accessed Jun. 14, 2021).

[43] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, EMF: Eclipse Modeling Framework 2.0, 2nd ed. Addison-Wesley Professional, 2009.

[44] I. van Gent, G. La Rocca, and L. L. Veldhuis, “Composing MDAO symphonies: graph-based generation and manipulation of large multidisciplinary systems,” in 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, American Institute of Aeronautics and Astronautics, 2017.

[45] KE-works, “KE-chain.” https://ke-chain.com (accessed Jun. 22, 2021).

[46] S. Chiesa, G. A. Di Meo, M. Fioriti, G. Medici, and N. Viola, “ASTRID--aircraft on board systems sizing and trade-off analysis in initial design,” Res. Educ. Aircr. Des., pp. 1–28, 2012.