32nd Annual INCOSE international symposium
hybrid event
Detroit, MI, USA
June 25 - 30, 2022

From System Architecting to System Design and Optimization: A Link Between MBSE and MDAO

Jasper Bussemaker
German Aerospace Center (DLR)
Hamburg, Germany
+49 (0)40 2489 641327
jasper.bussemaker@dlr.de

Luca Boggero
German Aerospace Center (DLR)
Hamburg, Germany
+49 (0)40 2489 641338
luca.boggero@dlr.de

Pier Davide Ciampa
German Aerospace Center (DLR)
Hamburg, Germany
+49 (0)40 2489 641322
pier.ciampa@dlr.de

Abstract. Optimization of system architectures can help deal with finding better system architectures in a large design space plagued by combinatorial explosion of alternatives. To enable architecture optimization, the design space should therefore be formalized into a numerical optimization problem, and it should be possible to quantitatively evaluate architecture alternatives. This paper presents a methodology for generating and modeling architecture design spaces using the Architecture Design Space Graph (ADSG), and using collaborative Multidisciplinary Design Analysis and Optimization (MDAO) techniques to evaluate architectures. Collaborative MDAO leverages disciplinary expertise while ensuring that analysis tools exchange data consistently and correctly using a central data schema. The problem solved in this paper is the missing link between architecture optimization and collaborative MDAO: the reflection of generated architectures in the central data schema. It is solved by the authors by mapping architecture components and Quantities of Interest (QOIs) to the central data schema using Data Schema Operations (DSOs). Such a mapping also assists the user in identifying missing or unnecessary disciplinary analysis tools. Three web-based software tools implementing the methodology are presented. Finally, the methodology and tools are demonstrated using the design of a supersonic business jet as an example.

Introduction

The developments presented in this paper are part of the EU-funded AGILE4.0 project[1], wherein the German Aerospace Center (DLR) is leading the development of a Model-Based Systems Engineering (MBSE) framework to enable the design of complex systems all the way from stakeholders and needs to detailed design using collaborative MDAO (Ciampa & Nagel, 2021). The MBSE framework developed in this context consists of the upstream architecting phase and the downstream product design phase, visualized in Figure 1. The upstream architecting phase consists of activities like identifying goals, specifying scenarios and requirements, and designing the system architecture.

[1] https://www.agile4.eu/

Typically, this phase is executed in a Model Based Systems Engineering (MBSE) context. The downstream product design phase consists of the selection of disciplinary tools needed for the design stage, the integration of these into a design process, and operation of the design system to design and optimize the system. These steps are typically executed in a Multidisciplinary Design Analysis and Optimization (MDAO) context.

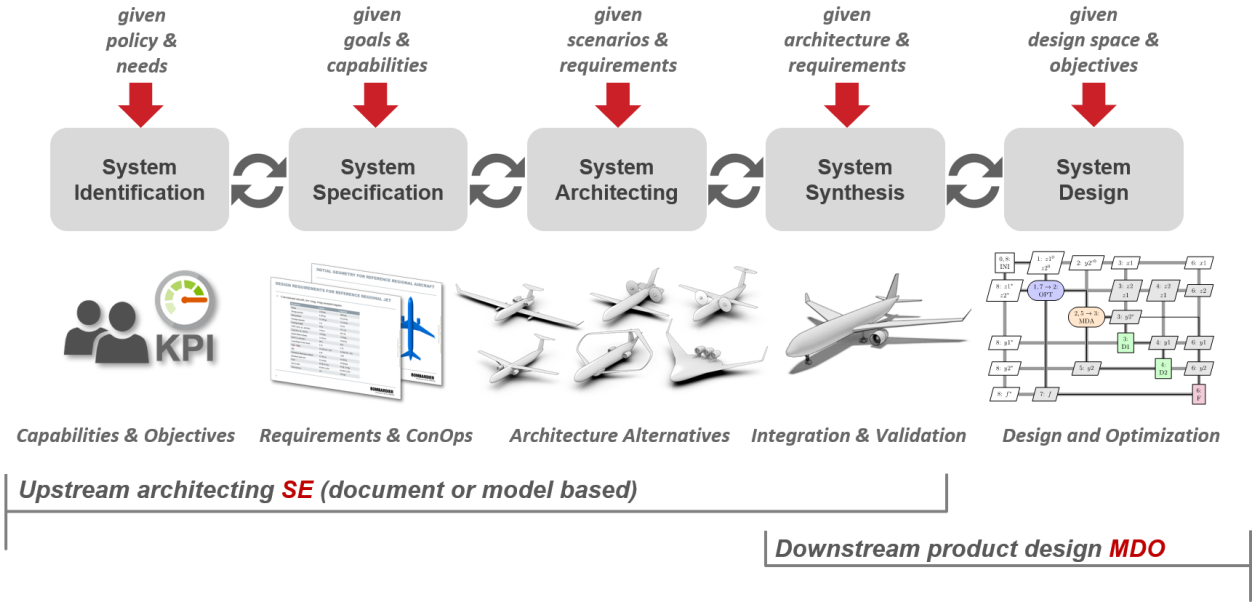

Figure 1. The MBSE framework developed in the AGILE4.0 project, showing the different phases and relation to systems engineering and MDAO. Reproduced from (Ciampa & Nagel, 2021).

The upstream architecting phase includes the design of the system architecture: a description of which components a system consists of, and how they work together to fulfill the system functions (Crawley, et al., 2015). The design of system architectures deals with an extremely large design space due to a combinatorial explosion of alternatives (Iacobucci, 2012). One way of dealing with this is by applying **systematic design space exploration of system architectures**, in practice meaning that optimization techniques are applied to the search for the best system architecture(s). To achieve this, the architecture design space should be modeled and formalized, and quantitative evaluation of architecture alternatives should be available (Bussemaker, et al., 2021).

Further downstream in the design process more detailed analysis is performed to size and optimize a design according to one or more design objectives. At this stage, the design of complex systems may involve several interacting and often conflicting engineering disciplines that all have to be considered to achieve at least a feasible design and at best an optimal design. Such disciplinary interactions can be automated by using MDAO techniques. Using MDAO, all relevant disciplines are treated simultaneously, ensuring that an analyzed design will be feasible between the disciplines (e.g. when the results of two analyses depend on the results of the other) and a design with appropriate compromises and synergies between the disciplines can be found (Sobieszczanski-Sobieski, et al., 2015).

Organizational aspects pose a major challenge to applying MDAO to the design of complex systems, especially when heterogeneous teams collaborate across organizational and even national boundaries (Ciampa & Nagel, 2020). To achieve integration at this level it is necessary to solve two problems: all disciplines should “speak the same language”, and it should be possible to exchange data across organizational boundaries while respecting intellectual property rights. These problems are tackled by the **collaborative MDAO** paradigm (see also Figure 2): a central data language (also known as central product model) is established that all disciplinary tools use as their input and output formats, and techniques are developed to enable data transfer using a central data server and requests for

execution. In aircraft design, the Common Parametric Aircraft Configuration Schema (CPACS) represents such a central data language (Alder, et al., 2020).

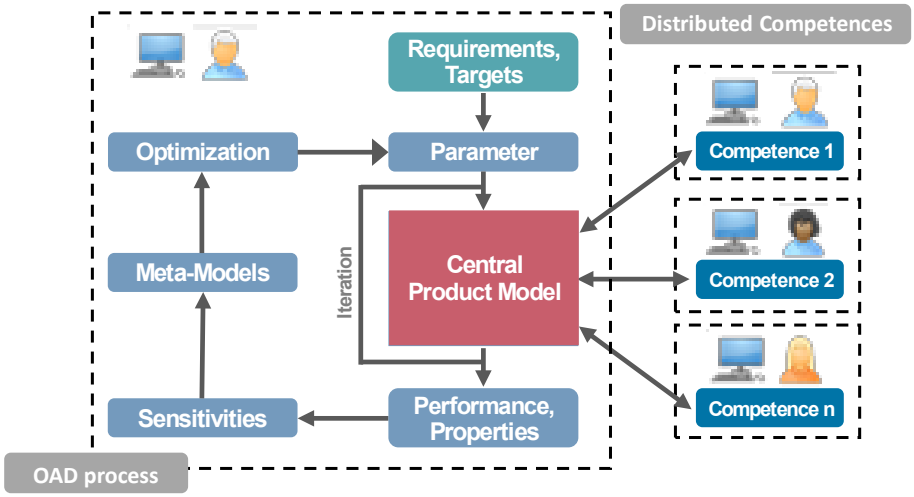

Figure 2. Collaborative MDAO framework, showing distributed competences (both code and expertise). Reproduced from (Ciampa & Nagel, 2016).

The core to enabling the interoperation of the upstream and downstream phases lies in bridging the MBSE and MDAO activities. As identified in (Ciampa & Nagel, 2021), specifically two bridges need to be established to do this:

1. **MBSE Specification to MDAO:** System requirements are related to the MDAO process.
2. **MBSE Architecting to MDAO:** Architecture components are related to the MDAO process.

The main function of both bridges is to verify that the developed MDAO design process sufficiently covers the requirements and architectures so that a realistic and useful result can be obtained from it. Additionally, these bridges can be used to automatically provide feedback as part of the architecture optimization loop and automatically verify requirements for a given design solution, similar to the strong coupling approach identified by (Chaudemar & de Saqui-Sannes, 2021).

The novel methodology presented in this paper focuses on the second bridge: linking MBSE architecting to MDAO to enable architecture optimization. It must however be noted that the architecting step builds on the requirements specification step, and therefore a large part of the requirements validation can also be taken up by the architecting activity through the specification of architecture properties and formulation of the architecture design problem. For example, performance requirements can be used as optimization constraints.

The rest of the paper is structured in four sections: first the underlying methodology is discussed, then the implemented tools for practically using this methodology are presented. Then, the methodology and tools are demonstrated using an optimization of a supersonic business jet as an example. Finally, the paper is concluded and an outlook is presented.

Methodology

To properly present the MBSE Architecting to MDAO bridging approach, first the architecting and MDAO approaches are presented in more details. The architecting approach focuses on modeling the function-based architecture design space for architecture optimization. The MDAO approach implements the collaborative MDAO principle to leverage disciplinary expertise and implement cross-organizational MDAO workflows.

Architecture Design Space Modeling

The first step to enabling architecture optimization is to model and formalize the architecture design space. In the developed MBSE framework this is done using the **Architecture Design Space Graph (ADSG)**: a graph-based formulation mapping functions to components and in addition representing component characterization and connection decisions. See Figure 3 for an example. It offers general applicability to and compatibility with MBSE methods, and due to its function-based nature it offers a method free of solution-bias for modeling architecture design spaces. Additionally, function-based architecting offers a natural connection from system requirements.

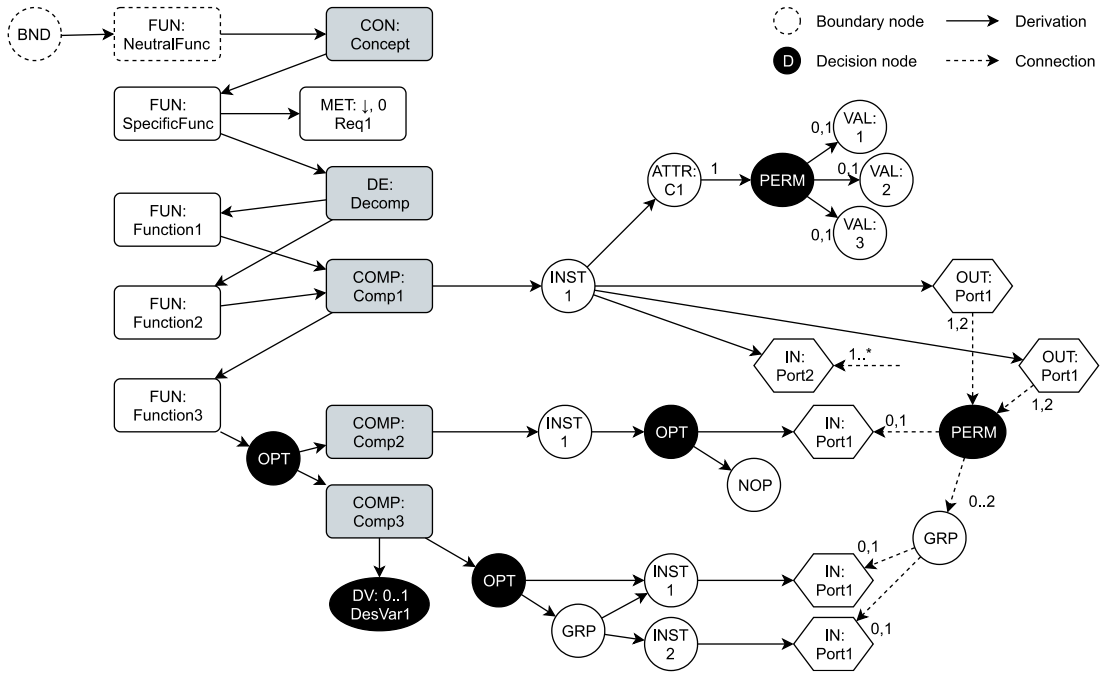

Figure 3. Example Architecture Design Space Graph (ADSG). Directed edges indicate derivation, decision nodes indicate a selection of mutually-exclusive options. Figure reproduced from (Bussemaker, et al., 2020).

An ADSG is constructed from a database of components that specify which functions they fulfill and which functions they need in order to do so (also known as function induction). Additionally, concepts, decompositions, and incompatibility constraints can be used as complexity management tools. Architectural decision nodes are automatically inserted for predefined patterns, for example if multiple components fulfill a function. Components then can represent several characterization choices: by number of instances and by attributes. Finally, component connection choices can be modeled using ports and permutation decisions. For more information about all the modeling elements and behavior of the ADSG, the interested reader is referred to (Bussemaker, et al., 2020).

An architecture optimization problem can be formulated from the ADSG by mapping decision nodes to design variables. Objectives and constraints are defined using **Quantities of Interest (QOIs)**: values associated to functions or components that can serve different roles during an optimization process. Next to objectives and constraints, QOIs can also be used as design variables (in addition to design variables defined from decision nodes), input parameters, and output metrics. QOIs may refer to performance requirements, thereby establishing a link between the requirements definition and architecting activities.

In this section we have shown how the ADSG might be used to model and formalize function-based architecture design spaces, and how an optimization problem can be formulated from it. The evaluation of architecture instances needed for enabling architecture optimization is not prescribed by the ADSG. In this paper an evaluation approach using collaborative MDAO will be presented,

however it must be noted that any evaluation method might be used (e.g. a set of custom Python scripts), as long as it can correctly simulate relevant effects and return some numerical values for the requested objectives and constraints given an architecture instance.

Collaborative MDAO

Collaborative MDAO encompasses several technologies needed for implementing an MDAO workflow in a cross-organizational context. In this section, however, the focus will lie on the use of a central data schema as this relates to how a product is parameterized and how disciplinary tools exchange data to converge to a solution. In aircraft design an example of an established central data schema is CPACS (Alder, et al., 2020). CPACS is XML-based and used by multiple research, academic, and commercial organizations (Ciampa & Nagel, 2020).

The involvement of many different engineering disciplines results in different jargon, product parameterizations, and units being used. The advantage of using a **central data schema** for product representation forces everyone to "speak the same language", thereby greatly reducing this problem. Another advantages is the reduction in the number of data interfaces that need to be implemented when creating an MDAO workflow (Alder, et al., 2020): from N(N-1) to 2N, where N is the number of disciplinary tools (see Figure 4). As a result, the development of an MDAO workflow is made truly collaborative as the data exchange at the tool level is managed by the owner of the disciplinary tool, whereas the data exchange between tools can then be managed by the workflow integrator (i.e. the person defining the workflow).

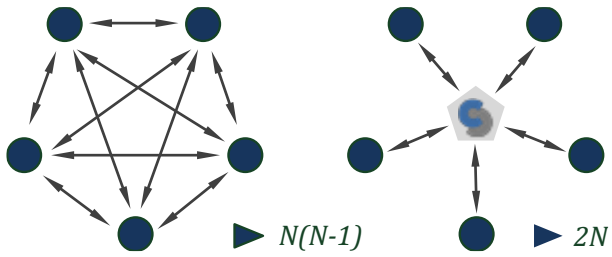

Figure 4. The use of a central data schema like CPACS reduces the number of implemented data interfaces from N(N-1) to 2N. Reproduced from (Alder, et al., 2020).

Determining which tools needs what data at what time is a challenging task when done manually, especially if the data also has to be guaranteed to be consistent in the presence of feedback loops. This task can be partly automated by using a graph-based method where information on tool interfaces (i.e. which subset of central data schema nodes are needed as input and provided as output) is used to represent data flows between disciplinary tools and the nodes of the data schema (van Gent, et al., 2017). This graph is used to query properties of the workflow, for example which data nodes are inputs, outputs, or collisions. Such information is used to determine data couplings between tools and the best order of execution for solving the workflow. Additional properties are specified to finally transform it to an MDAO workflow, for example assigning design variables, objectives, and constraints, and adding converger and optimizer elements.

The graph-based workflow model is then used to construct a workflow in a Process Integration and Design Optimization (PIDO) environment. This principle has been demonstrated for RCE [2], Optimus[3], and OpenMDAO[4] (van Gent, 2019). Although such PIDO environments also enable the manual definition of MDAO workflows, the great advantage of using the graph-based MDAO

[2] https://rcenvironment.de/
[3] https://www.noesissolutions.com/our-products/optimus
[4] https://openmdao.org/

workflow modeling method is that it is much easier to modify the workflow (e.g. add tools, rearrange tools) and to guarantee the correct data exchange at the same time.

In summary, the collaborative MDAO approach enables the integration of large-scale MDAO processes involving many heterogeneous engineering disciplines, by exchanging data using a central data schema. This forces all disciplines to speak the same language, results in the implementation of less interfaces, and enables disciplinary experts to focus on their tool while the process integrator can focus on the definition of the MDAO workflow. Collaborative MDAO can be a powerful method to designing and optimizing complex engineering products, however research on how to use it for architecture optimization is in its very early stages, see for example (Jeyaraj, et al., 2021).

Connecting Architecting and Collaborative MDAO

The previous sections have introduced the methodologies for architecture design space modeling and optimization and for composing collaborative MDAO workflows. To apply the latter for evaluating the performance of generated system architectures, it is needed to establish a two-way connection. First of all, the connection from generated architectures to collaborative MDAO, the feed-forward connection, should enable the synchronization between the architecture definition and the central data schema instance. Then, after the MDAO workflow has analyzed a particular architecture, a feedback connection should be established to extract performance data (i.e. optimization objectives and constraints) from the central data schema and communicate them to the architecture optimizer.

Another function of the architecture-to-MDAO bridge is to support the selection of the disciplinary tools during the workflow definition process (Ciampa & Nagel, 2021). From the architecture design space model, it is known which components and associated Quantities of Interests (QOIs) the architecture design space consists of. From the MDAO workflow model, it is known which disciplines use which central data schema nodes as their input or provide them as their output. By mapping the components and QOIs to the central data schema, it can then be deduced which disciplines are associated to which architecture components. This knowledge can assist the system integrator in several ways:

1. **Verify** that the selected disciplines sufficiently cover the architecture components.
2. Identify **interchangeable** (i.e. redundant) disciplinary tools: multiple tools associated to the same set of components.
3. Identify **missing** disciplinary tools: components not associated to any disciplinary tools.
4. Identify **unnecessary** disciplinary tools: tools not associated to any component.

Together, these knowledge aspects help justify the inclusion of disciplinary tools and ensure that the tools selected for evaluating architecture performance sufficiently cover the architecture elements. Additionally, it provides a step towards building confidence in the evaluation process providing useful and realistic results.

From the above discussion it can be seen that mapping components and QOIs to the central data schema both assists the system integrator with the selection of the engineering disciplines and enables the feed-forward connection of synchronizing architecture instances with the central data schema. This mapping is established by defining a so-called **Data Schema Operation (DSO)** for each QOI and/or component. An example can be writing (reading) a QOI value to (from) some data node.

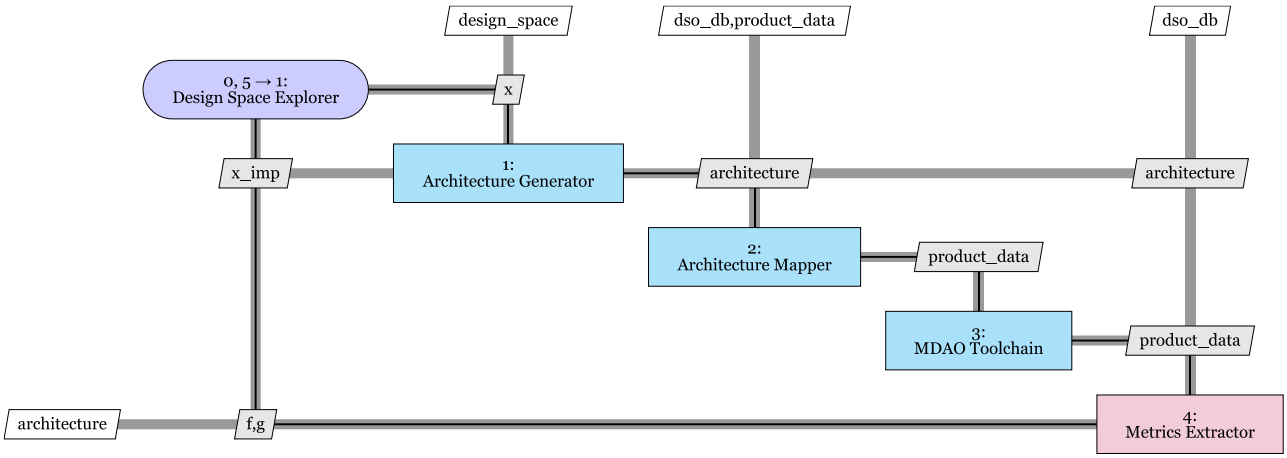

Figure 5. Optimization loop with architecture evaluation: the design space explorer (i.e. optimizer) suggests a design vector (x) that is converted into an architecture description. Generated architectures are mapped to a central data schema instance (product_data) using the data schema operation database (dso_db), which is then used to execute the collaborative MDAO toolchain.

Defining these data schema operations also enables integration into the architecture optimization loop, as notionally shown in Figure 5. Collaborative MDAO for architecture evaluation is implemented using four blocks:

1. **Architecture generator:** generates an architecture from the design vector (the vector of all design variables, as suggested by the design space explorer), taking design variable hierarchy into account (for more information, refer to (Bussemaker, et al., 2020)).

2. **Architecture mapper:** takes a generated architecture, an initial product model using the central data schema, and the database of Data Schema Operations (DSO database), and produces a product model representing the generated architecture.

3. **MDAO toolchain:** the actual collaborative MDAO workflow taking the product model as input and producing an updated (i.e. analyzed) product model as an output.

4. **Metrics extractor:** reads relevant metrics from the product model according to the DSO database and updates QOIs of the generated architecture to reflect MDAO output.

The end result of the architecture optimization loop then consists of an architecture instance with correctly updated QOIs and data schema instance representing the same architecture. Due to the automated mapping connections these two different system representations are guaranteed to be consistent.

Integration of the architecture mapper and metrics extractor blocks in the MDAO workflow itself is also possible, because each DSO knows which data schema node it affects: combining all DSOs together then yields a complete view of all nodes affected by the architecture mapper and all nodes needed for metrics extraction.

Implementation

This section presents the implementation as done by DLR in the EU-funded AGILE4.0 project. The three elements needed for implementing the presented architecture optimization methodology are implemented in three web-based software tools. Their backends are programmed in Python[5], their

[5] https://www.python.org/

frontends are built using the Vue[6] framework. Their web-based nature enables access without the need for installing anything locally, and it enables automated synchronization with a central database to store all relevant models and data for some design project.

Architecting Design Space Modeling: ADORE

The architecture design space modeling method based on the Architecture Design Space Graph (ADSG) has been implemented in a tool called ADORE (Architecture Design and Optimization Reasoning Environment) by the DLR. ADORE implements the following functionalities:

1. ADORE project file format for storing the design space definition, design problems, and generated architecture instances;
2. **Constructing an ADSG** including architectural decisions from the design space definition;
3. Creating **architecture instances** from an ADSG by manually assigning an option to each decision;
4. Defining **design optimization problems** from an ADSG and using this to generate architectures from design vectors;
5. Interfaces to several **optimization frameworks**, including OpenMDAO and pymoo[7];
6. Application Programming Interface (API) for implementing **architecture evaluation**.

This setup means that the user is not directly interacting with the ADSG while using ADORE. Rather, the model being manipulated is the same as the ADORE project file format, where a project consists of a `DesignSpace`, zero or more `DesignProblem` instances, and zero or more `Architecture` instances. The detailed presentation of the ADORE project format is out of scope for this paper.

The architecture evaluation API enables custom evaluation code on a per-problem basis. The only functionality to implement is to provide numerical values for requested QOIs for a given `Architecture` instance. ADORE takes care of correctly generating architectures from the design vector and feeding back the provided numerical values to the optimization algorithm as objectives and constraints.

The graphical user interface of ADORE has been developed to enable interactive editing of the architecture design space model, as shown in Figure 6. Feedback is provided continuously enabling quick recovery of errors and greater insight into model behavior. A list of architectural decisions can be viewed to verify that all decisions are implemented correctly. Design space model behavior can be verified by manually creating architecture instances: options are selected for each architectural decision, taking decision hierarchy into account (i.e. inactive decisions won't be shown). The user can then verify that by taking decisions indeed the expected architecture instances can be generated.

[6] https://vuejs.org/
[7] https://pymoo.org/

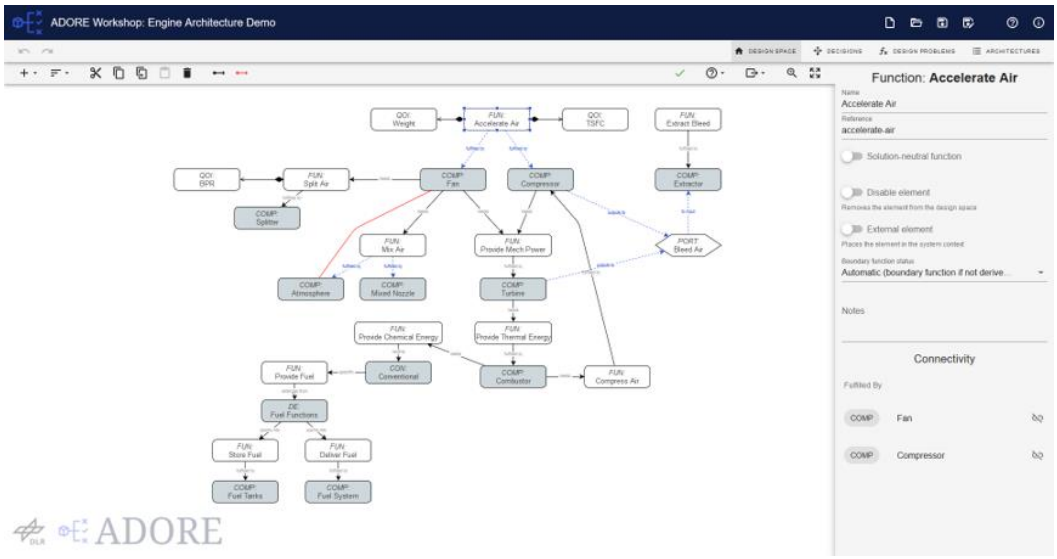

Figure 6. ADORE web-based graphical user interface showing the design space editing canvas.

The user interface also enables the formalized definition of design optimization problems: architectural decisions are mapped to design variables, and QOIs are assigned roles including additional design variables, objectives, and constraints. Design variables can be fixed to some specific value to modify the size of the optimization design space, and the roles of objectives and constraints can be modified to change optimization problem behavior. These features are useful to for example define a simpler architecture optimization problem for testing the optimization toolchain, or to try out different optimization objectives and compare their results.

Figure 7 shows a part of a design space model for a jet engine architecting problem. Derivation of architectures starts at the boundary function “provide propulsive power”. Blue-dashed lines represent architecture decisions, for example whether to add a fan or not, and whether to use a mixed nozzle or not. Red lines represent incompatibility constraints, meaning that elements on both ends cannot exist together in an architecture instance. This specific model has been used to solve a jet engine architecting problem: engine performance was evaluated using the framework presented by (Bussemaker, et al., 2021), which was connected to ADORE using the architecture evaluation API.

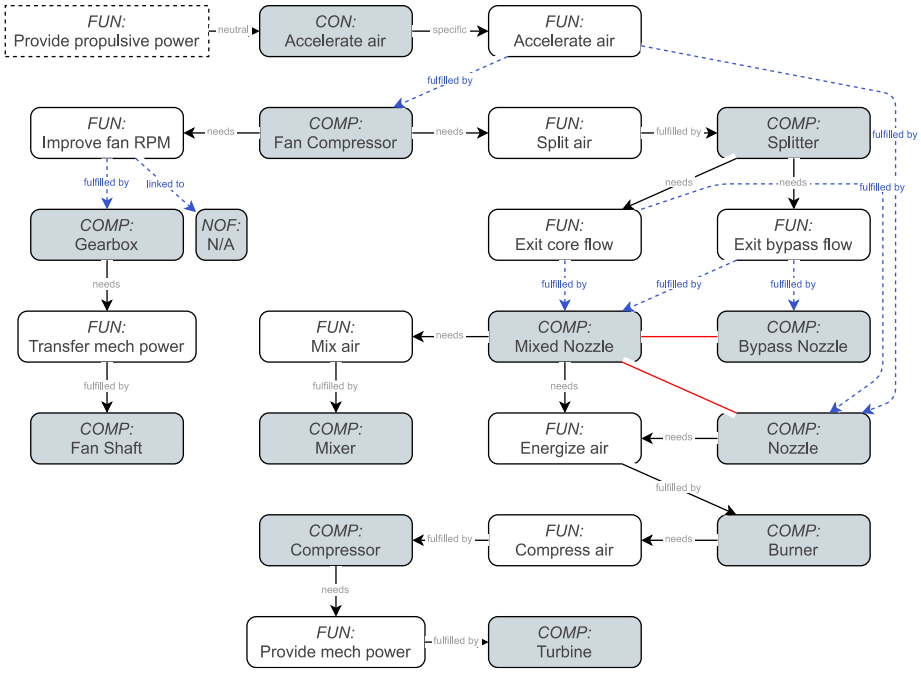

Figure 7. Partial view of a jet engine architecture design space model in ADORE.

MDAO Workflow Modeling: MDAx

MDAO workflows are modeled using MDAx (MDAO Workflow Design Accelerator), a tool developed by the DLR. MDAx implements the previously presented collaborative MDAO workflow modeling methodology, and does so using an interactive user interface. This makes it more intuitive for users to create workflows, because they can see the immediate effect on data flow that manipulations have, and also enables the use of the program in a more exploratory setting, for example during workshops or for documentation of processes. For example, Figure 5 has been created using MDAx. For more details regarding MDAx and its design philosophy, please refer to (Page-Risueño, et al., 2020).

Due to brevity, no more details regarding MDAx will be included in this paper. Here it suffices to mention that workflows modeled in MDAx can be exported to the CMDOWS format (van Gent, et al., 2018) or directly to an executable RCE workflow for execution.

Linking Architecting to MDAO: MultiLinQ

As discussed before in the methodology section, it is clear that the two main functions of the tool linking architecting to MDAO are to implement **Data Schema Operations (DSOs)** and to assist in the selection and verification of disciplinary tools. These functions are implemented by the authors in a tool called MultiLinQ. MultiLinQ enables the definition of components and QOIs and the selection of multidisciplinary tools according to their input and output definitions.

One the one hand, MultiLinQ enables the definition of components and QOIs. QOIs may be associated to components or they may be standalone QOIs, for example when directly associated to some performance requirement. Components and QOIs may come from various sources; it is for example possible to simply import an ADORE project and extract components and QOIs from there. This way additionally the link between components and QOIs in the ADORE and MultiLinQ projects are established which eases integration in the architecture optimization loop.

On the other hand, tool input and output definitions can be added to define the data schema and relate nodes to disciplinary tools. Here too this data can come from multiple sources, for example from a database of input and output definitions exported from MDAx.

To connect components and QOIs to the data schema, a DSO is configured for each component and QOI which defines how it influences the data schema instance. Additionally, in case of the QOI it is also specified whether it represents an input value or output value, which determines when it will be applied: in the architecture mapper (for input) or in the metrics extractor (for output), see also Figure 5. Components and QOIs can exist in multiple instances in a generated architecture, which should also be possible to be reflected in the central data schema. Table 1 shows possible DSOs for components and QOIs and different number of instances.

Table 1: Possible Data Schema Operations (DSOs).

Architecture Element	**DSO [number of element instances]**		
	[0]	**[1]**	**[2 or more]**
QOI (input)	Write empty value; Remove node; Do nothing	Write value to node	Write list of values; Write to copied nodes
QOI (output)	N/A	Read value from node	Read from list; Read from copied nodes
Component	Remove node; Do nothing; Write nr of instances	Create new node; Write nr of instances	Copy (new) node; Write nr of instances

The mapping from components and QOIs to disciplinary tools can be visualized in the **Component-Tool (CT) matrix**. Figure 8 shows an example CT matrix. It can be seen how components and associated QOIs are associated to disciplinary analysis tools, for example the "Reference area" QOI of the "Wings" component is associated to the "Aerodynamics" and "Structures" tools. The CT matrix also shows unmapped elements. In the presented example, it shows that the "Fuel price" QOI of the "Fuel system" component is not mapped to any tool, indicating a missing tool. Also, the "Cost" tool is not associated to any component or QOI, indicating an unnecessary tool for the problem at hand.

		Tools				
Components	QOIs	Aerodynamics	Cost	Performance	Propulsion	Structures
Engines	Base engine weight				✓	
Engines	Thrust				✓	
Fuel system	Fuel price					
Fuel system	Misc fuel weight					✓
Wings	AR	✓				✓
Wings	Reference area	✓				✓
Wings	Sweep	✓				✓
Wings	Taper ratio					✓
Wings	t/c	✓				✓
Wings structure	Load factor					✓
Wings structure	x					✓
	Altitude	✓		✓	✓	

Figure 8. Component-Tool (CT) matrix showing how components and QOIs are associated to disciplinary analysis tools.

The main usage scenario of MultiLinQ is then to first define components, QOIs, tools, and DSOs, which in practice will be an iterative process between ADORE, MultiLinQ and MDAx. Once that has been completed, the architecture optimization can be started. This can either be done by giving ADORE control over the optimization loop, or by integrating ADORE and MultiLinQ as executable blocks inside the PIDO environment running the MDAO toolchain. The advantage of ADORE control is that state-of-the-art optimization algorithms can be selected, and that the combination of optimization algorithm, ADORE, and MultiLinQ can be executed on a different computer than the MDAO workflow. This would enable a scenario where ADORE and MultiLinQ are offered as an online service, so that users only have to concern themselves with running the MDAO workflow used for architecture evaluation and nothing else.

Demonstration: Optimization of a Supersonic Business Jet

The presented methodology for performing architecture optimization using collaborative MDAO is demonstrated using the supersonic business jet design problem from (Sobieszczanski-Sobieski, et al., 1998). The collaborative MDAO implementation is provided here [8], and includes both tool implementations and associated input and output definitions.

The optimization problem represents a classical multidisciplinary aircraft design problem: a coupled design of structures, aerodynamics, propulsion, and performance. The MDAx model is shown in Figure 9 on the left: aerodynamics, propulsion, and structures are coupled due to feedback connections, whereas the performance calculation is a post-processing tool that does not provide

[8] https://github.com/DLR-SL-MDO/mdax-ssbj

feedback to any other tool. Design variables include wing planform parameters (aspect ratio, reference area, sweep, etc.), mission parameters (cruise altitude, Mach number), and engine rated thrust. Calculation outputs include weights (maximum take-off weight, fuel weight, engine weight, etc.), cruise lift and drag, structural stresses, and wing twist. The performance tool calculates the range and endurance, both calculated using the Breguet equations. The workflow is executed in RCE, an open-source PIDO environment developed by the DLR. The right side of Figure 9 shows the workflow as exported to RCE, ready for execution.

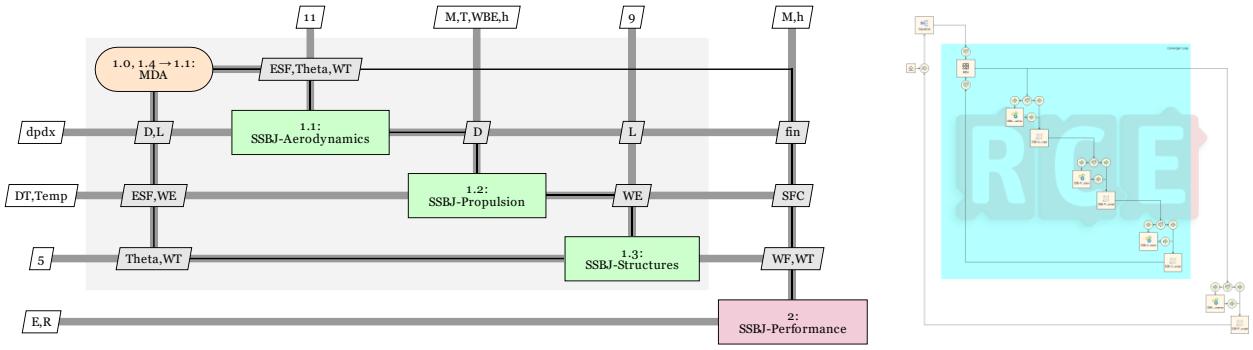

Figure 9. XDSM view (left) and RCE workflow (right) of the collaborative MDAO workflow for analyzing one supersonic business jet configuration, modeled in MDAx.

The architecture design space model of the supersonic business jet consists purely of QOIs representing design variables and calculation outputs of the original MDAO problem. It should be noted, however, that the methodology presented in this paper also works for architectural choices regarding the selection of components. The architecture design space model is shown in Figure 10. It is built-up from the boundary function "Transport payload", as would come from upstream requirements. This function is decomposed into "Generate lift" and "Generate thrust", which are fulfilled by their respective components. QOIs are associated to the boundary function, representing system-level QOIs, and to the components. For example, the right side of Figure 10 shows QOIs associated to the "Engines" component. Not shown in the design space model visualization are QOI roles: for example, design variable QOIs have bounds associated, input parameters have a static value, and constraints have their reference value and a direction (i.e. greater than or lower than).

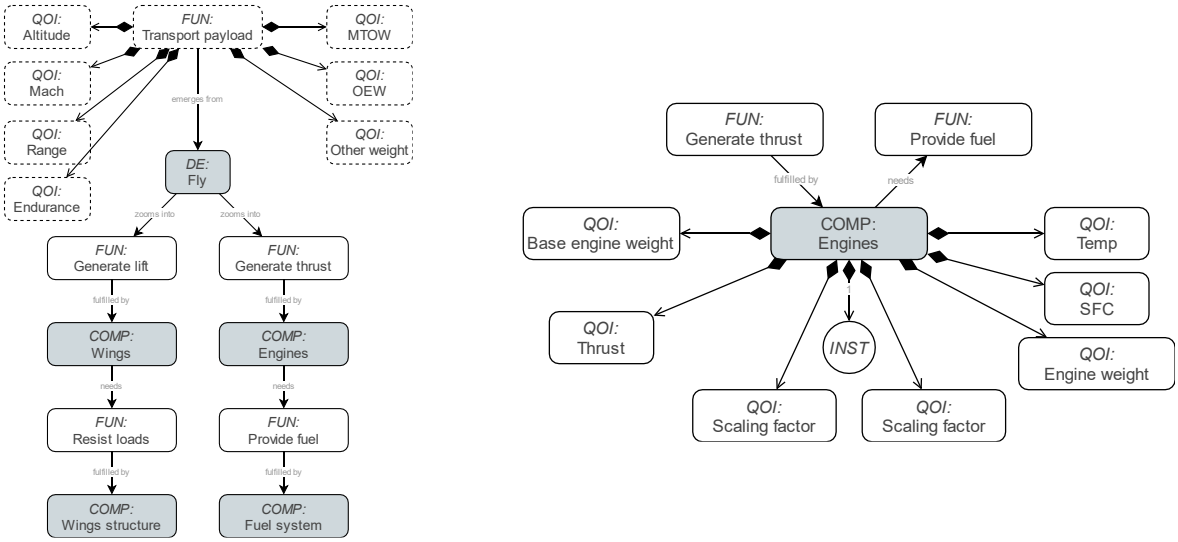

Figure 10. ADORE architecture design space model of the supersonic business jet, showing the system overview on the left, and "Engines" component details view on the right.

At this point, it is clear that on the one side the MDAx MDAO workflow model enables the quantitative evaluation of a supersonic business jet model, and that ADORE represents the system

architecture model as would come from upstream MBSE activities. To bridge this gap, MultiLinQ is used to assign Data Schema Operations (DSOs) to QOIs. In this demonstration, only the reading and writing of QOI values are used as DSO. The resulting CT matrix is shown in Figure 11. It shows that all QOIs are mapped to an associated MDAO discipline, and that all disciplines are necessary. Many QOIs only associate with one discipline (e.g. all engine-related QOIs), whereas some are associated to multiple disciplines, for example because multiple disciplines need their value to do their calculations.

		Tools			
Components	**QOIs**	SSBJ-Aerodynamics	SSBJ-Performance	SSBJ-Propulsion	SSBJ-Structures
Engines	Base engine weight			✓	
Engines	Thrust			✓	
Fuel system	Misc fuel weight				✓
Wings	AR	✓			✓
Wings	Reference area	✓			✓
Wings	Sweep	✓			✓
Wings	Taper ratio				✓
Wings	t/c	✓			✓
Wings structure	Load factor				✓
Wings structure	x				✓
	Altitude	✓	✓	✓	
	Mach	✓	✓	✓	
	Other weight				✓
Engines	Engine weight			✓	
Engines	SFC			✓	
Engines	Scaling factor			✓	
Engines	Scaling factor			✓	
Engines	Temp			✓	
Fuel system	Fuel weight				✓
Wings	D	✓			
Wings	L	✓			
Wings	L/D	✓			
Wings	dpdx	✓			
Wings structure	Stress1				✓
Wings structure	Stress2				✓
Wings structure	Stress3				✓
Wings structure	Stress4				✓
Wings structure	Stress5				✓
Wings structure	Twist				✓
Wings structure	Twist				✓
	Endurance		✓		
	MTOW				✓
	OEW			✓	
	Range		✓		

Figure 11. MultiLinQ Component-Tool (CT) matrix for the supersonic business jet problem.

The optimization loop is executed under control of ADORE. In this case, the pymoo interface to ADORE is used to connect an optimization algorithm. The control flow representing this optimization loop is presented in Figure 12. It shows that ADORE is in charge of generating the architecture, MultiLinQ is used for mapping the architecture to/from the input/output files (i.e.

implementing both the “architecture mapper” and “metrics extractor” steps), and RCE is used to execute the collaborative MDAO workflow. Files are transferred from ADORE to RCE using a custom server implementation by ADORE, here called “remote server”. Note that the optimization framework (here: pymoo), workflow execution environment (here: RCE), and file transfer mechanisms (here: remote server) are all interchangeable, for example by OpenMDAO, Optimus, and Brics (Moerland, et al., 2020) respectively. This setup would also enable an architecture-optimization-as-a-service setup, where the optimization algorithm, ADORE, and MultiLinQ run on a server, sending input files to the user’s computer for evaluation.

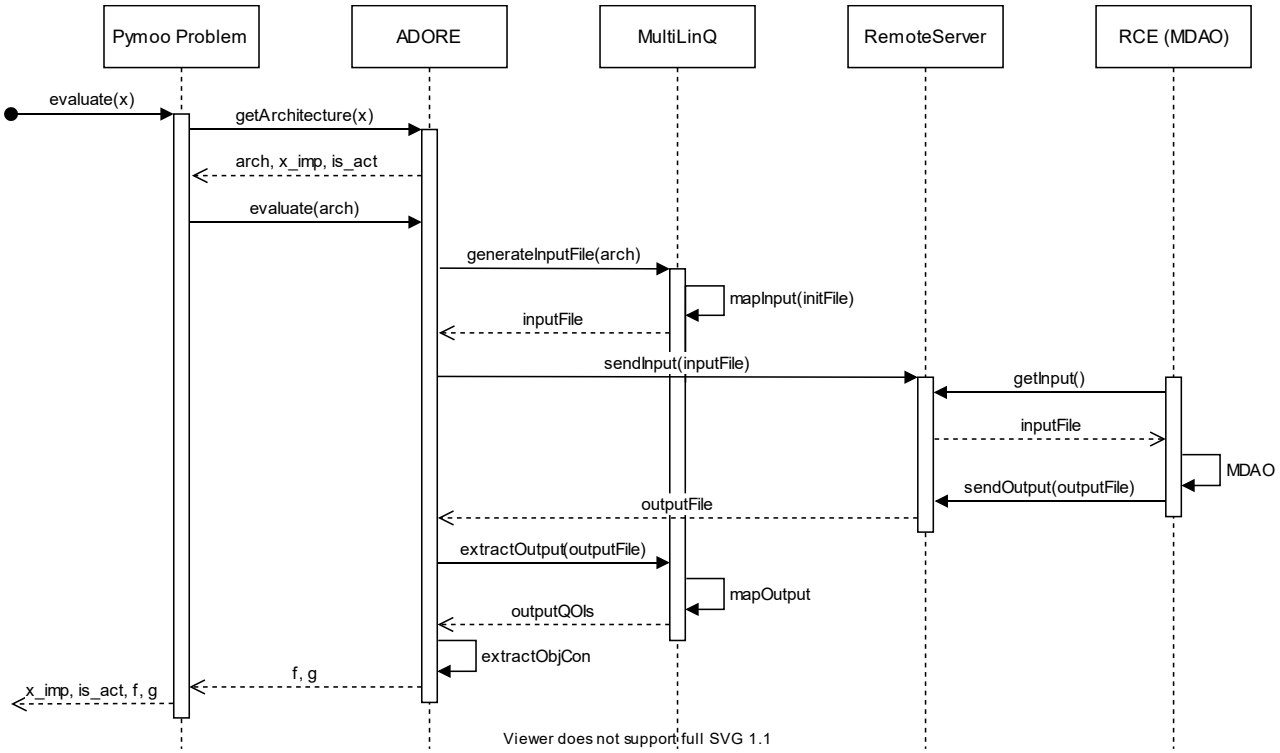

Figure 12. Sequence diagram showing what happens when a design vector (x) is evaluated in the optimization loop. The remote server implements the mechanism for sending the input file from ADORE to RCE, where the collaborative MDAO workflow is executed.

The optimization problem is executed using a Surrogate-Based Optimization algorithm with a Kriging surrogate model (Bussemaker, et al., 2021). A bi-objective formulation is used: both range and L/D are attempted to be maximized. Figure 13 displays the resulting Pareto front, showing that the presented framework can be used to solve a realistic architecting and collaborative MDAO problem.

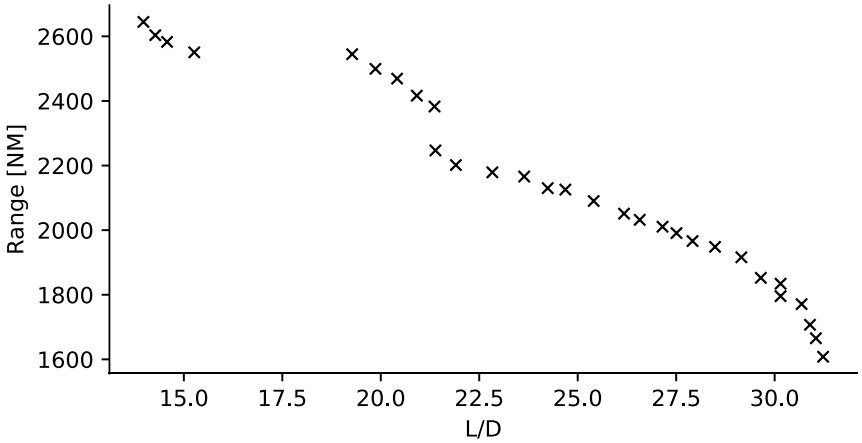

Figure 13. Pareto front of the multi-objective supersonic business jet optimization problem.

Conclusion and Outlook

A methodology for modeling and executing architecture optimization problems using collaborative MDAO for architecture evaluation is presented. The methodology uses the Architecture Design Space Graph (ADSG) for function-based modeling of the architecture design space and formulating the associated hierarchical, mixed-discrete, multi-objective optimization problem. Collaborative MDAO uses a central data schema to communicate data between disciplinary analysis tools, thereby enabling the implementation of MDAO workflows leveraging diverse engineering expertise across organizational boundaries. The architecture optimization method and collaborative MDAO are then linked by mapping Quantities of Interest (QOIs) to the central data schema using Data Schema Operations (DSOs). From this mapping a Component-Tool (CT) matrix can be generated, which supports the user in justifying and selection of disciplinary tools for solving the architecture optimization problem at hand.

The three elements of the methodology are implemented in three web-based tools: ADORE for architecting, MDAx for collaborative MDAO workflow modeling, and MultiLinQ for linking architecting and collaborative MDAO. MDAx is used to create workflows in RCE where it can be guaranteed that data exchange happens exactly as specified in the workflow model. ADORE and MultiLinQ are used to create an optimization problem and connect it to an optimization framework of choice.

A supersonic business jet optimization problem is used to demonstrate the methodology. The optimization loop is structured such that the chosen optimization framework has control over execution, and the generated input file is sent to RCE using the remote server implementation of ADORE. It is shown that the presented methodology and tools combine into a feasible and usable methodology to create architecture optimization problems using collaborative MDAO for architecture evaluation.

Next developments will mainly focus on further development of MultiLinQ and the underlying methodology. It will be investigated whether the DSOs suggested in this publication will be sufficient for enabling architecture optimization that also includes function fulfillment choices (i.e. whether components will be included in the architecture or not), and component characterization and connection choices.

The methodology will be applied to several industry-provided aircraft design application cases in the context of the AGILE4.0 project, which includes application cases focusing on supply chain management, electrification of an existing aircraft, retrofitting, and maintenance system design. Design problems include MDO systems with up to ten coupled tools and tens of architectural decisions (leading to millions of possible architectures).

Acknowledgments

Part of the research presented in this paper has been performed in the framework of the AGILE 4.0 project (Towards cyber-physical collaborative aircraft development) and has received funding from the European Union Horizon 2020 Programme under grant agreement n. 815122. The authors are grateful to the project partners for their insightful inputs and feedback about the developments presented in this paper.

References

Alder, M., Moerland, E., Jepsen, J. & Nagel, B., 2020. *Recent Advances in Establishing a Common Language for Aircraft Design with CPACS.* sl, sn

Bussemaker, J. H., Ciampa, P. D. & Nagel, B., 2020. *System Architecture Design Space Exploration: An Approach to Modeling and Optimization.* sl, American Institute of Aeronautics and Astronautics.

———, 2021. *Effectiveness of Surrogate-Based Optimization Algorithms for System Architecture Optimization.* sl, American Institute of Aeronautics and Astronautics.

———, 2021. *System Architecture Design Space Modeling and Optimization Elements.* Shanghai, sn

———, 2021. *System Architecture Optimization: An Open Source Multidisciplinary Aircraft Jet Engine Architecting Problem.* Virtual Event, American Institute of Aeronautics and Astronautics.

Chaudemar, J.-C. & de Saqui-Sannes, P., 2021. *MBSE and MDAO for Early Validation of Design Decisions: a Bibliography Survey.* sl, IEEE.

Ciampa, P. D. & Nagel, B., 2016. *Towards the 3rd Generation MDO Collaborative Environment.* Daejeon, sn, p. 1–12.

———, 2020. AGILE Paradigm: The next generation of collaborative MDO for the development of aeronautical systems. *Progress in Aerospace Sciences,* November.Volume 119.

———, 2021. *Accelerating the Development of Complex Systems in Aeronautics via MBSE and MDAO: a Roadmap to Agility.* Virtual Event, American Institute of Aeronautics and Astronautics.

Crawley, E., Cameron, B. & Selva, D., 2015. *System architecture: strategy and product development for complex systems.* sl:Pearson Education.

Iacobucci, J. V., 2012. *Rapid Architecture Alternative Modeling (Raam): a Framework for Capability-Based Analysis of System of Systems Architectures,* sl: sn

Jeyaraj, A. K., Tabesh, N. & Liscouet-Hanke, S., 2021. *Connecting Model-based Systems Engineering and Multidisciplinary Design Analysis and Optimization for Aircraft Systems Architecting.* sl, American Institute of Aeronautics and Astronautics.

Moerland, E. et al., 2020. Collaborative Architecture supporting the next generation of MDAO within the AGILE paradigm. November, Volume 119, p. 100637.

Page-Risueño, A., Bussemaker, J. H., Ciampa, P. D. & Nagel, B., 2020. *MDAx: Agile Generation of Collaborative MDAO Workflows for Complex Systems.* sl, American Institute of Aeronautics and Astronautics.

Sobieszczanski-Sobieski, J., Agte, J. & Sandusky, R., 1998. *Bi-level integrated system synthesis (BLISS).* Reston, American Institute of Aeronautics and Astronautics, p. 164–172.

Sobieszczanski-Sobieski, J., Morris, A. & van Tooren, M. J. L., 2015. *Multidisciplinary Design Optimization Supported by Knowledge Based Engineering.* sl:John Wiley & Sons, Ltd.

van Gent, I., La Rocca, G. & Veldhuis, L. L., 2017. Composing MDAO symphonies: graph-based generation and manipulation of large multidisciplinary systems. *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference.*

van Gent, I., La Rocca, G. & Hoogreef, M. F. M., 2018. CMDOWS: a proposed new standard to store and exchange MDO systems. *CEAS Aeronautical Journal,* Volume 0, p. 0.

van Gent, I., 2019. *Agile MDAO Systems: A Graph-based Methodology to Enhance Collaborative Multidisciplinary Design,* sl: sn

Biography

Jasper Bussemaker. Jasper received his MSc in Aerospace Engineering with a focus on aircraft design and MDO from Delft University of Technology in 2018. Currently he researches system architecture optimization and MDO at the DLR Institute of System Architectures in Aeronautics in Hamburg, Germany. He is developing methods for modeling architecture design spaces, and for coupling architecture optimization to collaborative MDO.

Luca Boggero. Luca obtained in 2018 his PhD in Aerospace Engineering at Politecnico di Torino with a dissertation on Multidisciplinary Design and Optimization (MDO), Model Based Systems Engineering (MBSE) and design of aircraft subsystems. He now works as a Research Scientist at the DLR Institute of System Architectures in Aeronautics in Hamburg, and he leads the System Integration & MDO Group. He coordinates and is involved in research projects within the context of MDO, Systems Engineering and MBSE.

Pier Davide Ciampa. Pier leads the MDO team at the DLR Institute of System Architectures in Aeronautics. He is actively researching and leading projects in the field of aircraft design and systems architecting, Multidisciplinary Design and Optimization (MDO) and Model Based Systems Engineering (MBSE). In 2018 he received the ICAS Award for Innovation in Aeronautics for his work in the AGILE project. Currently he coordinates the EU Horizon 2020 project AGILE4.0.