MDAx : ENHANCEMENTS IN A COLLABORATIVE MDAO WORKFLOW FORMULATION TOOL

Sparsh Garg[1], Jasper H. Bussemaker[1], Luca Boggero[1] & Björn Nagel[1]

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

Abstract

The design of a complex product like an aircraft involves different engineering disciplines, which are coupled with each other and exchange data in terms of inputs and outputs. Integration of such disciplines, their interaction with each other and exchange of diverse datasets leads to a time and resource intensive workflow development process. The interconnections between the coupled disciplines can lead to design inconsistency and transparency loss. To address these issues, the MDAO Workflow Design Accelerator (MDAx) was developed. MDAx bridges the gap between the competence deployment phase and multidisciplinary design analysis and optimization (MDAO) workflow execution by offering a comprehensive MDAO workflow modeling platform. It is designed to be quick to learn and user-friendly to operate, and provides efficient methods to inspect and explore the different disciplinary components of the workflow and their data couplings. MDAx uses the eXtended Design Structure Matrix (XDSM) representation to describe the workflows and takes a step further with some additional operations resulting in an executable workflow which can be exported to process integration platforms. This paper presents the latest additions to the environment, like modelling sub-workflows as components, supporting dynamic MDAO, value based identification of variables, and separate execution tool naming. These developments are made based on feedback from various collaboration partners, in order to enhance the ease of adoption of collaborative MDAO paradigms by competence developers without requiring expert knowledge in MDAO architecting. Several application cases that leveraged the capabilities of MDAx in the past few years are presented to demonstrate the potential of the software.

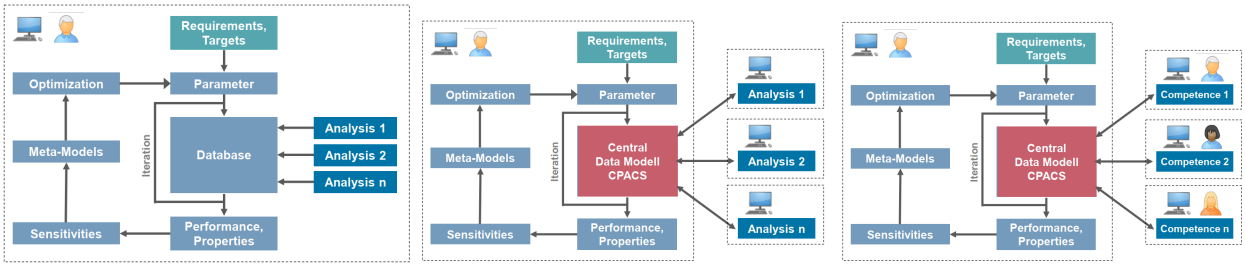
Keywords: Multidisciplinary Design, Analysis and Optimization; Digital Aircraft Development

1. General Introduction

The process of designing aircraft is inherently multidisciplinary and encompasses the definition and creation of tightly coupled analysis systems. As aerospace systems continue to grow in complexity, there is a need for the advancement of design approaches that make use of models and physics-based simulation. These approaches aim to seamlessly integrate system design within a framework of Model-Based Systems Engineering (MBSE) [1, 2]. MBSE approaches can help accelerate the development of aircraft products through proper setup, deployment and operation of Multidisciplinary Design Analysis and Optimization (MDAO) [3, 4] systems. The reason behind adoption of MDAO is to analyse not only the individual disciplines but also the relationships and interaction between them. The advantage of using MDAO techniques can be seen in terms of better designs at lower time and cost penalties [5]. To that end, the AGILE Paradigm was a novel approach introduced to MDAO processes, helping in reducing the setup time of MDAO systems to more than 40%, with respect to conventional MDAO approaches [6].

The AGILE paradigm was a result of MDAO systems being evolved over the past years from being restricted to single integrator - monolithic design systems to a multi integrator - polylithic system which can consist of many disciplinary tools. This evolutionary process can be seen through the three **generations of MDAO** [7], as depicted in Figure 1. The "1st generation" focused on the

integration of the optimizer into the single large system consisting of all the disciplinary capabilities. Although, this made the design process more time efficient, it left the desire for a distributed and scalable system which enabled dynamic exchange and update of the integrated disciplinary system. The "2^{nd} generation" overcame the scalability aspect of the analysis part of the system, to a certain extent, by utilizing high performance computational facilities to distribute large-scale analysis tasks. Such tasks could now be performed quicker by relaying them to the respective computational facility at the central optimizer's request. The challenges that remained were the presence of a single integrator for the whole design system and of the increasing number of systems, and their respective disciplines. These were addressed with the "3^{rd} generation" MDAO system where the distribution of tasks went beyond computational analysis facilities, to include all the engineers, distributed among multiple specialized organizations [6]. Collaborative Engineering methods are used to realise such "3^{rd} generation" MDAO system.



(a) 1^{st} Generation: Integrated MDAO Systems

(b) 2^{nd} Generation: Distributed MDAO Systems

(c) 3^{rd} Generation: Collaborative MDAO Systems

Figure 1 – Evolution in MDAO sytems over the years (L→R). The dotted lines represent various computational domains linked within the MDAO system [7]

Collaborative Engineering is a domain of engineering where disciplinary stakeholders work together to achieve a common goal. It focuses on the interaction of these stakeholders with each other to integrate their expertise in order to realise their mutual interests and derive joint outcomes. By combining the different strengths of multiple disciplines, the collaborative engineering process aims to achieve a more comprehensive and efficient outcome [8].

An important step into the process of collaborative engineering is to bring all the disciplines, along with their disciplinary experts, under a single umbrella and providing them with data-driven and computer supported tools and methodologies to consolidate their expertise, ideas and knowledge in an attempt to achieve a well-organized, multi-disciplinary, multi-engineer, and multi-organization development process [9].

Since many different disciplines collaborate together, the utilization of a common data model significantly reduces the number of possible interconnections between them and ensures a consistent source of information[10]. Using a Central Data Schema (CDS) ensures that all disciplines "speak the same language" and enables implementation and reuse of disciplines in collaborative MDAO workflows. An example of a CDS is CPACS [10], an open-source XML-based format for exchanging aircraft design data which has been applied successfully in many projects across disciplines and organizations. Setting up workflows based on a CDS is supported by workflow formulation platforms that automatically detect data connections based on input-output definitions of disciplinary tools.

To that end, the MDAO Workflow Design Accelerator (MDAx), developed by the Institute of System Architectures in Aeronautics at the German Aerospace Centre (DLR), Hamburg, aims to facilitate collaborative MDAO processes. MDAx provides stakeholders with an intuitive, interactive platform for modeling MDAO problems using eXtended Design Structure Matrices (XDSMs). This paper discusses recent advancements in MDAx, enhancing its capability to address complex and novel MDAO problems, and revisits the foundational methodology behind the tool. Additionally, it highlights applications of MDAx, showcasing examples from global organizations who have utilized the platform for their MDAO studies.

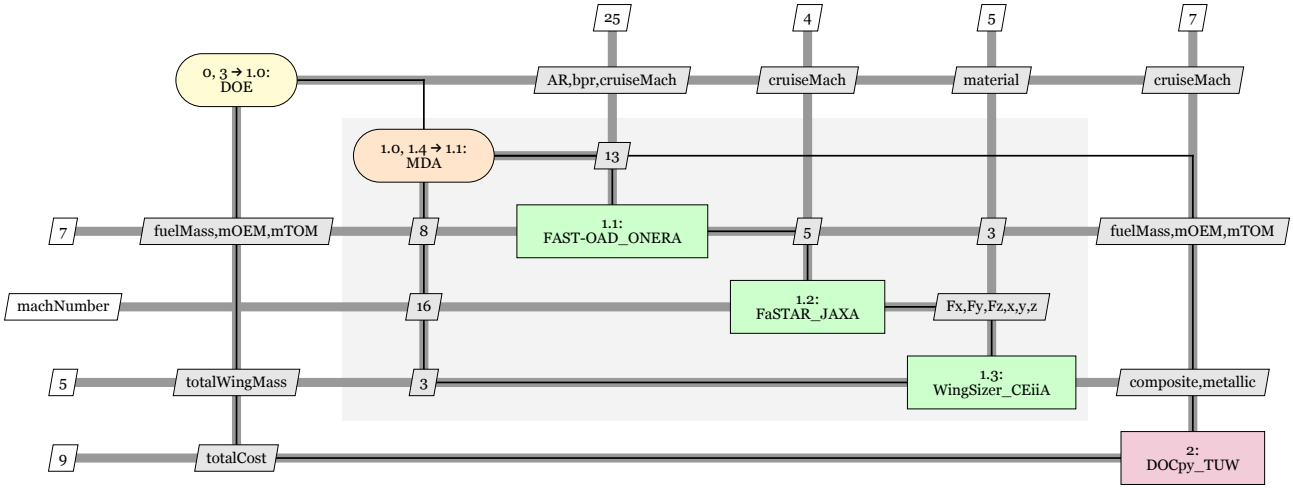
2. Methodology and Current Approach

Several MDAO problem formulation tools have been developed in the past, like OpenMDAO [11], KADMOS [12] and GEMSEO [13]. Now, while some of them try to provide a platform that can address the additional features of collaborative engineering, which is required for the modern-day efficient aircraft design process, all of them take a *"top-down approach"* to MDAO problem formulation. The top-down approach refers to the fact that designer needs to have the final MDAO problem well defined with all the competences, their input and output (I/O) specifications and their couplings with other competences, before it can be formulated in an MDAO platform. This is a prerequisite for the working of the aforementioned tools. MDAx, on the other hand, employs a more *"bottom-up approach"* to the formulation of the MDAO problem. The bottom-up approach gives more autonomy and independence to the integrator and competence developers from the very beginning. MDAx eliminates the necessity for designers to have pre-existing knowledge of the final MDAO problem formulation. It provides the users with a very simple and intuitive interface. This is one of the major benefits of using MDAx over other MDAO platforms. It enables the designers to inspect, add, and modify data couplings between competences and introduce new competences without requiring a complete re-iteration of the MDAO process. Furthermore, it utilizes a CDS, which is fundamental to the collaborative MDAO process. A distinguishing feature of MDAx is the transparency it provides regarding the structure of the CDS file and the node creation and usage, allowing users to inspect and understand the couplings of their analysis tools. This transparency enables users to make necessary adjustments to the CDS or coupling connections before executing the workflow, thereby facilitating the transition from traditional MDAO to collaborative MDAO. This transition supports the adoption of methodologies such as the AGILE paradigm in the industry [6]. MDAx specifically addresses the design process formulation phase of the AGILE paradigm by automating the MDAO workflow development which bridges the gap between the upstream competence deployment phase and the downstream workflow execution phase. The resistance of the industry to adopt the AGILE paradigm was largely due to the limitations like inflexibility in the setup processes, difficult customization of workflows, and lack of user-friendliness [14]. This was addressed, to a large extent, via the MDAx tool.

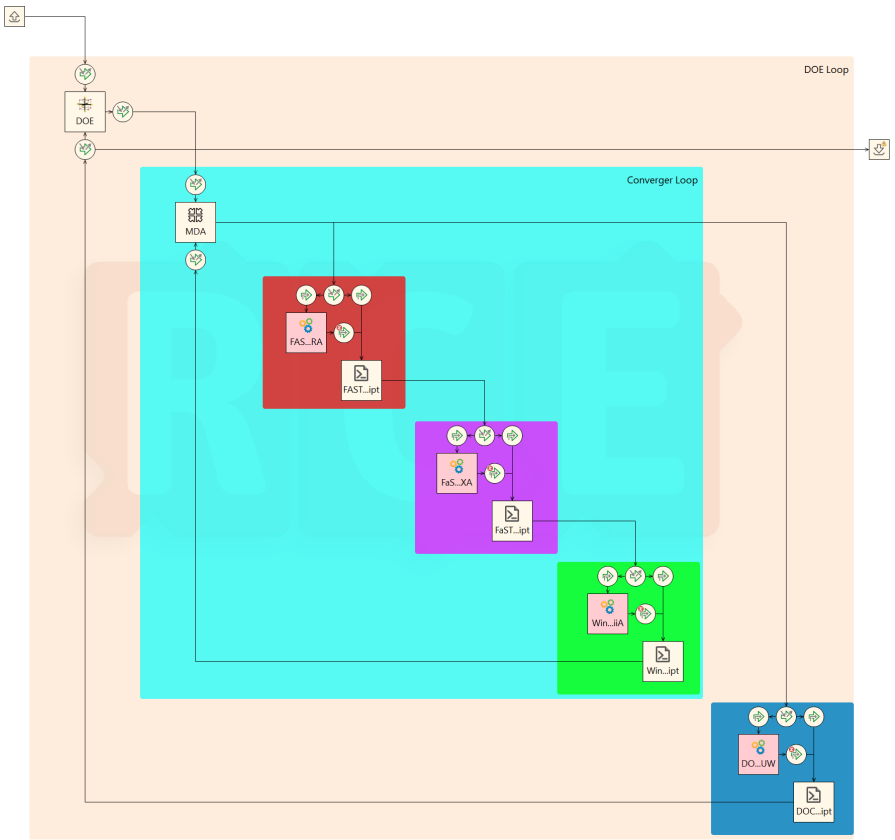
MDAx, offers a novel approach to streamline collaboration in large-scale engineering projects. Instead of pre-defining an MDAO problem and structuring the workflow accordingly, MDAx allows workflow integrators to directly interact with an XDSM interface. This interaction involves drag-and-drop operations, providing immediate feedback and the ability to undo actions, all without being bound by rigid procedures. This grants users the flexibility to efficiently create and refine their workflow models, representing tool configurations and process logic for executable workflows. There's no need for pre-existing schemes or tool repositories to initiate the modeling process. These can be defined as the project progresses, facilitating the customization of workflows to suit specific preferences. This allows for the exploration of potential workflow configurations that users may not have initially considered. While conventional MDAO architectures can be generated if required, they are not rigidly mandated for the creation of executable workflows.

The user interface (UI) is centered around the XDSM visualization format, providing a standardized workflow model that allows for a clear examination of tool interactions. This includes real-time evaluation of variable collisions and feedback connections. Details about this and the theory behind them are discussed by Page Risueño et. al. [15]. The UI is intuitive and user-friendly, requiring minimal introduction and explanation, thus reducing the learning curve. Export capabilities to various Process Integration and Design Optimization (PIDO) tools facilitate the generation and organization of MDAO workflows prior to execution in the preferred framework, minimizing manual setup efforts.
MDAx serves as an MDAO workflow model editor, diverging from a rigid step-by-step approach to enhance usability. It offers a space for domain experts and integrators to explore and grasp the overall system without the need for specific methodologies or technical jargon. The goal is to instill confidence in engineers to navigate across various domains while comprehending the complexity of the entire system, ultimately streamlining collaborative MDAO processes.

In other words, MDAx helps close the implementation gap between formulating the workflow based on MDAO solution strategy and the execution of the workflow in a PIDO platform [16]. Figure 2a shows a XDSM built for a sustainable aircraft design problem where different partner tools, and the



(a) Initial part of a XDSM built for a sustainable aircraft design usecase with different IFAR partners contributing their tools which interact in a collaborative MDAO framework with each other



(b) A portion of the executable workflow export for RCE with visible tool blocks along with their pre and post processing blocks

Figure 2 – MDAO workflow model for design of a sustainable aircraft as part of the IFAR-ECN 2023 Conference [8]

coupling of variables between them, can be seen. The MDAO problem was part of the International Forum for Aviation Research (IFAR) - Early Career Networking (ECN) conference event called the IFAR-X Challenge [8]. The figure also shows an example of a converger driver being used to resolve feedback between partner tools. There is also a clear distinction made between tools with only input coupling (red block) as compared to tools with both input and output coupling (green blocks). A detailed description of XDSM representation can be found in [17]. A full MDAx workflow made for the IFAR-X event can be found in Appendix A. Experts from 18 IFAR member countries participated in

the challenge. Further details on the collaboration and the results of the design study are available in [8]. Figure 2b gives a view into the executable workflow which was exported from MDAx for an open-source execution environment developed within DLR called Remote Component Environment (RCE) [18]. The figure depicts the execution pathway that the simulation would follow and the direction of exchange of data. It can also be seen that a simple looking tool in the XDSM representation, in reality has a complex execution structure, full with pre and post processing scripts.

3. Recent Developments of MDAx

Extensive use of MDAx in aircraft design optimization studies in the years since its first release have shed light on important workflow features that were still missing from the platform. Having addressed some of them, it is the goal of the new developments to supplement the MDAO problem formulation capabilities of the tool. These features are discussed in this section.

3.1 Sub-workflow as Components

One such feature which was added was of using **'Sub-workflow as Components'**. Disciplinary experts usually have a set of competence tools that they commonly use and re-use in their analysis and optimization studies. MDAx already provides a feature which allows the user to save their competence inputs and outputs in XML format to later reintroduce the competence in another MDAO problem without re-specifying the associated data I/O connections. However, the sub-workflow feature takes it a step further by allowing, what can be referred to as *"nested workflows"*. Figure 3 further illustrates this concept. The component 'B' is a smaller workflow further consisting of two competences in itself. This smaller workflow is referred to as a Sub-workflow. It is an independent workflow which can be used as a component in other bigger workflow where its functionality is needed. Now, instead of manually copying and pasting, the user may instead opt to integrate the workflow shown in the figure as a tool to be used in other workflows. Another advantage of this feature is that it improves the visibility of the XDSM components. The XDSMs is made more readable by clustering the smaller and more detailed analysis components in a single component. Furthermore, it is possible to have multiple layers of nested workflows, essentially, sub-workflows within sub-workflows. These feature becomes even more powerful if the PIDO platform being used for execution of this workflow also supports the sub-workflow as component feature. This is the case with RCE which allows the competence experts to hide the details of the implementation, for example the use of an optimizer, from users of the component and to easily update that implementation.

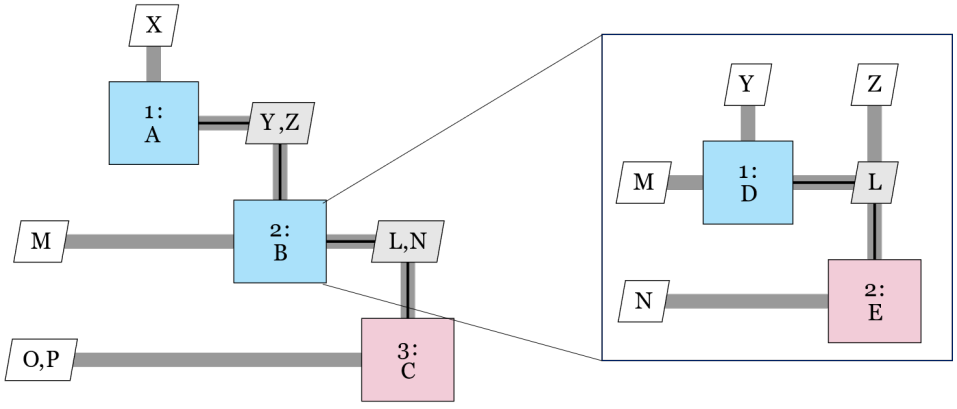


Figure 3 – Depiction of the sub-workflow element 'B' in the XDSM which consists of two discipline, namely, 'D' and 'E'. The I/O data of the sub-workflow elements are collectively represented as I/O of element 'B' for coupling with other disciplines of the outer workflow.

3.2 Dynamic MDAO

An MDAO workflow is formulated only after the system architecture for the system under investigation is defined. System architecting refers to identifying components that system consists of, and how

these components are combined to achieve the system goals [19]. However, a practical design study begins with a larger number of possible system architectures. To that end System Architecture Optimization (SAO) is employed for the application of numerical optimization algorithms to automatically explore the architecture design space [20].

One hurdle in SAO is that a separate MDAO formulation is required for each architecture which needs to be evaluated. This architecture specific workflow formulation and management is a time consuming and repetitive task. To formulate and execute MDAO problems for application in SAO, MDAO platforms have to deal with multiple challenges that arise during MDAO workflow development, due to involvement of multiple objectives, hierarchical variables, mixed-discrete coupling variables and black-box disciplines [21].

This section summarises the recent efforts to enable the use of collaborative MDAO techniques for SAO [22].

Methodology

When addressing different system architectures, it is clear that modifications to the MDAO problem formulation are necessary, tailored to the specifics of each new architecture. This section would use existing examples available in literature to further discuss this fact. However, the most common modifications to the MDAO problem formulation and execution, as caused by the different system architectures, can be accommodated into four mainstream categories which are called ***architectural influences***:

1. **Conditional variables:** Each system is made of different components, and each component is defined by different variables. As a consequence, the variables existing in the MDAO problem will change depending on the architecture being analysed. The variables whose existence depends on an architectural decision are called conditional variables.

2. **Data connection:** In some occasions, the connections between design disciplines might change because of an architectural decision. This happens when there is a modification in the coupling of a certain variable leading to a change in terms of which disciplines the variable is connected to. Therefore, in MDAO problems used for system architecture optimization, rerouting of variables can happen, leading to dynamic connections between disciplines.

3. **Discipline repetition:** In some MDAO problems a discipline has to be repeated a fixed number of times. However, in system architecture optimization, sometimes the discipline multiplicity is a design variable and not a predefined fixed value, and therefore has to be readjusted automatically.

4. **Discipline activation:** As a consequence of an architectural decision, there are multiple cases where some of the disciplines included in the MDAO problem are no longer required for that specific architecture. This is usually the case when two or more technologies are available to perform a certain function.

Supporting Architectural Influences in MDAx

MDAO problems are formulated in several steps [16]: a tool repository is defined, the MDAO problem is defined by selecting tools from the repository and establishing data connections, and the solution strategy is defined by adding non-linear solvers and/or applying an MDAO architecture [5]. To benefit from MDAO for SAO problems, another step needs to be added to include the architectural influences. These architectural influences are included by specifying the "influence logic": the logic (if statements, loops, etc.) which defines how the MDAO workflow should modify its behavior for evaluating different architecture instances. After formulation, the MDAO problem is deployed in some execution environment, where it can be invoked as part of the SAO loop for a given architecture instance generated by an architecture generator [21]. It should be noted that the formulation phase and the optimization loop form two coupled yet separate steps of the whole process, as also depicted in Figure 4. Four high-level strategies are identified for using MDAO for architecture evaluation: Single

static, Multi static, Single Dynamic and On-demand [22]. MDAx adopts the **single dynamic MDAO problem** strategy, which implements influence logic directly into the workflow such that its behavior is automatically modified for a given architecture instance during the execution phase (Figure 4).

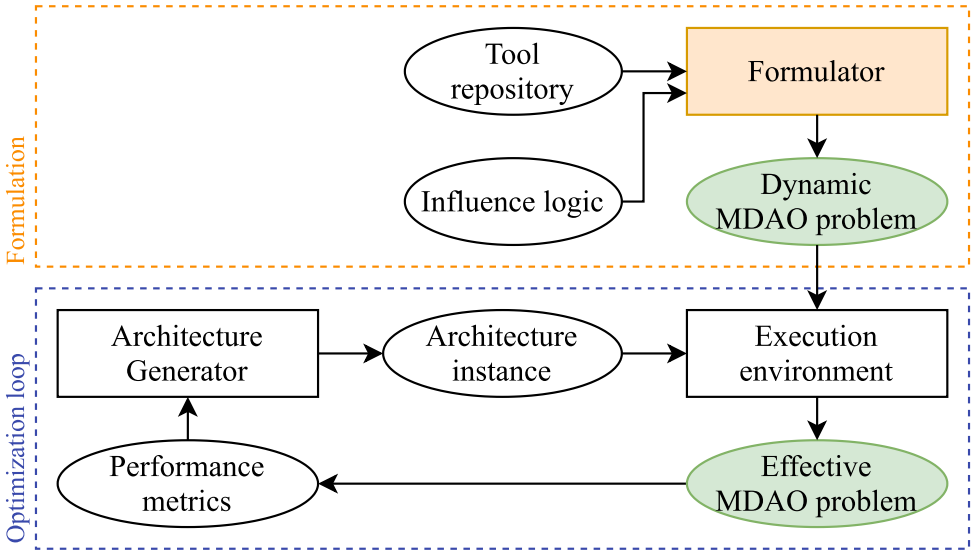


Figure 4 – High-level strategy for integrating architectural influences into MDAO workflows involves executing the formulation phase (indicated by orange dashes) prior to the optimization loop (indicated by blue dashes). The optimization loop then runs automatically without requiring user interaction [22].

MDAx has been modified so that the MDAO workflow model contains all the logic for dynamically modifying its execution behavior, which is materialized in an export to an external integration platform, RCE, where the SAO problem is executed. The implementation of the four architecture influences has been done as follows:

Discipline Activation: Discipline activation is implemented by attaching an activation assertion to each discipline dependent on an architectural decision. If the assertion is true, the discipline is executed; if false, it is skipped. These assertions are linked to specific nodes in the XML workflow file and are executed by a script in RCE. Conditions can be based on node existence, repetition, or node information. Assertions are implemented as composable Python classes that determine the routing of the input file through or around the tool.
Discipline Repetition: Discipline repetition can be configured based on two strategies: parallel (multiple instances) or series (single instance repeated). MDAx and RCE use the series configuration, translating multiple component instances into repeated nodes in the XML workflow file. Each execution involves converting data in the global XML workflow file to local representation and back, to avoid confusion between the dataset values for the different component instances. This is managed by "Global to Local" and "Local to Global" blocks. An "Iterator" block tracks iterations and ends the loop after the specified repetitions.
Data Connection: Data connections in MDAx, which uses a central data schema, are managed by modifying input-output specifications. Connections are deactivated by disabling the input variable in the discipline's definition, using assertions similar to those for discipline activation. These assertions filter out corresponding XML nodes if the condition is true.
Conditional Variables: MDAx handles conditional input variables by searching for potential input nodes in the XML workflow file before execution. Found nodes are added to the tool input file, while missing nodes are ignored. For conditional output variables, deactivation conditions specified in the configuration file are checked after tool execution. Satisfied assertions result in the variable being removed from the outputs and not merged into the workflow XML file.

It is possible to have some or all of the architectural influences in a workflow. Figure 5 shows an

example of the RCE components using which the four architectural influences are implemented. The dynamic MDAO developement was tested out on a space rocket design application case [22]. Further details of the study are out of the scope of this paper, however, the resultant XDSM for the problem can be seen in Figure 13 in Appendix B. The UI implementation of the dynamic MDAO feature is currently being implemented and therefore is not described in this publication.

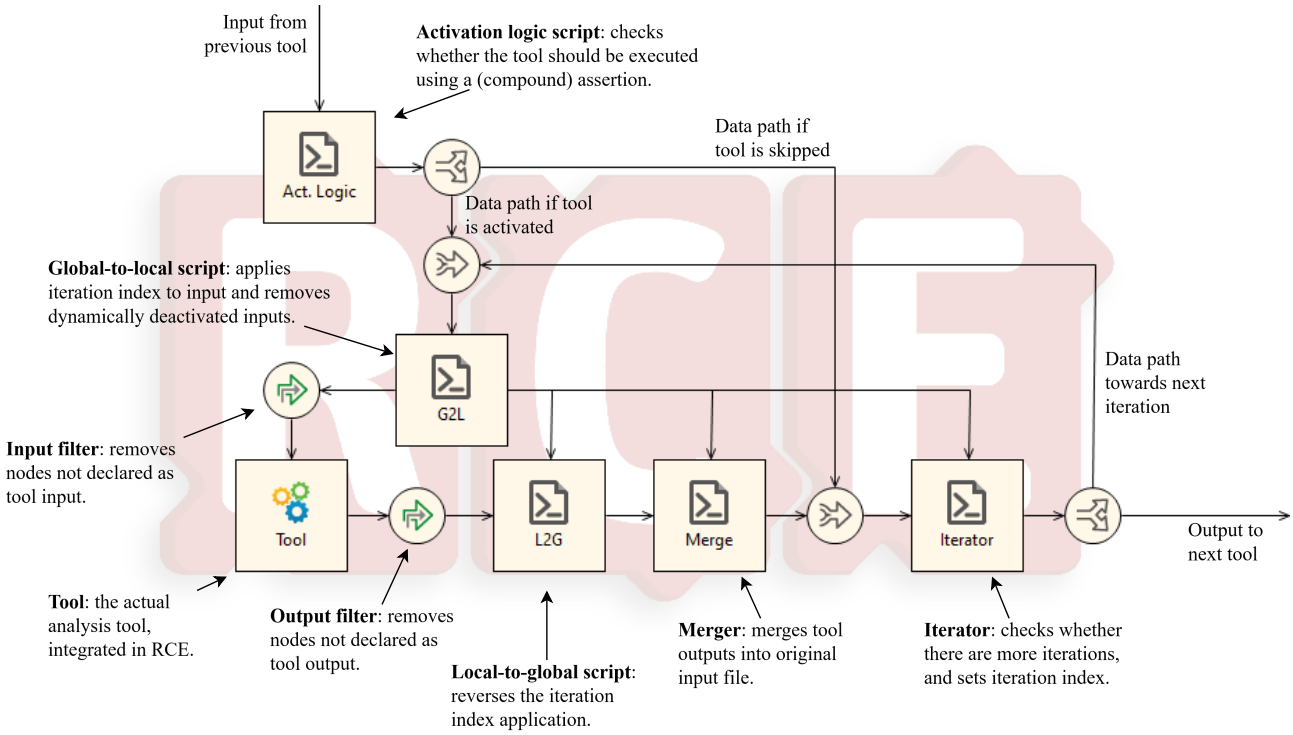


Figure 5 – An export from MDAx to RCE showing the procedure used to implement architectural influences in MDAx. First, the workflow XML enters into the activation logic script, which checks if the activation logic assertion attached to the tool is true or false. The Global to Local component handles various tasks such as determining the number of iterations, tracking the current iteration, and selecting inputs and outputs for each iteration. The Local to Global component prepares tool outputs for correct placement in the workflow XML. Lastly, the iterator block assesses whether the discipline repetition has concluded. Figure reproduced from [22].

3.3 Other added features

Another feature which was introduced was of separate **execution tool naming**, to differentiate it from the name of the competence component. It means that the user can now use the same executable tool in the background for different disciplinary components. For example, under the propulsion discipline, an expert can essentially use similar tools for motor and generator sizing where the major difference lies in the component specific inputs and outputs. MDAx provides a mapping script before the execution of the back-end simulation tool which maps the tool I/O from the variable structure being used in the collaborative MDAO problem. As long as the coupling of the I/O variables is consistent with choice of the tool being used for a generator sizing component or a motor sizing component, the same back-end tool can be used for both by specifying a common execution tool name which is different from either of the component names.

Another development by the name of **"sub-value matcher"** is introduced within the tool. This feature allows the user to differentiate between identically named variable nodes by making a value based distinction between them. Note, that the value under question must belong to a sub-node of the variable node to be distinguished. This feature was added based on a requirement for CPACS that arose during an overall aircraft design apllication-case under EXACT project [23], which is discussed in section 5. This feature is particularly helpful when the chosen CDS, CPACS in this case, has provision to have multiple nodes with the same name but which differ in the values of the sub-

nodes. In such a situation, sub-value matching can be used to specify the difference between these nodes by specifying the differentiating sub-node and their corresponding values, at the parent node level itself.

The implementation of the graphical user interface for these feature can be seen in the next section.

4. UI Implementation

MDAx's foundation lies in a Python library that handles all the logic behind its workflows. The user interface (UI) of MDAx is built on top of this code. It employs a web-based architecture: a Python script on the back-end starts a web server and sets up the core, while on the front-end, a Vue.js[1] application manages what the user sees and responds to their input. The back-end and front-end communicate through a websocket. The application can be deployed and then accessed as a service through any standard browser, without requiring installation on the user's local computer. However, it can also be deployed as a standalone application if necessary.

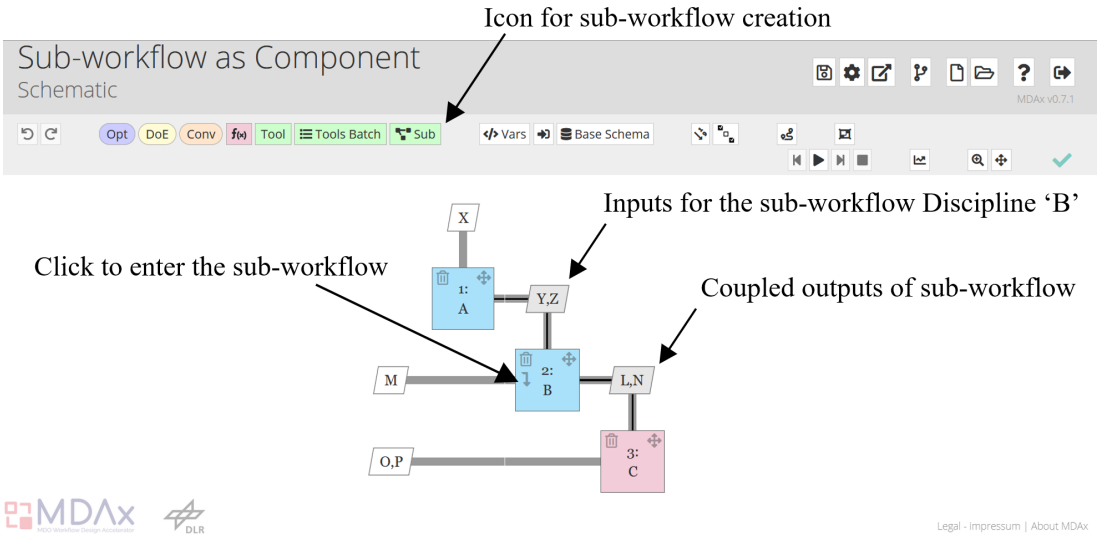
Figure 6 provides a view of the main UI screen: the workflow is presented in the center, with buttons above it for manipulating the workflow and managing the current project. The status indicator informs the user whether the workflow is currently executable. It can display a red exclamation mark (indicating issues) or a green check mark (signifying that the workflow is executable, as shown in the figure). More details on the basic UI of MDAx are well defined by Page Risueño et al. in the original MDAx paper [15].

For the implementation of the sub-workflow as component feature, another button was added in the tool bar as show in Figure 6a. All the inputs and outputs of the components within the sub-workflow are indicated as I/O specification of the sub-workflow component in the original workflow. The sub-workflow component also include an arrow icon to take the user inside the sub-workflow in order to edit it's contents. This icon is located on the top left of the component below the 'Delete' icon. The sub-workflow window is similar to the original window with exception of the arrow in the title bar which can be used to navigate to the outer workflow. Also, the sub-workflow component creation icon is still visible in this window due to the fact that MDAx allows for multi-level nesting of workflow. Thus the user can create sub-workflows within existing sub-workflows.

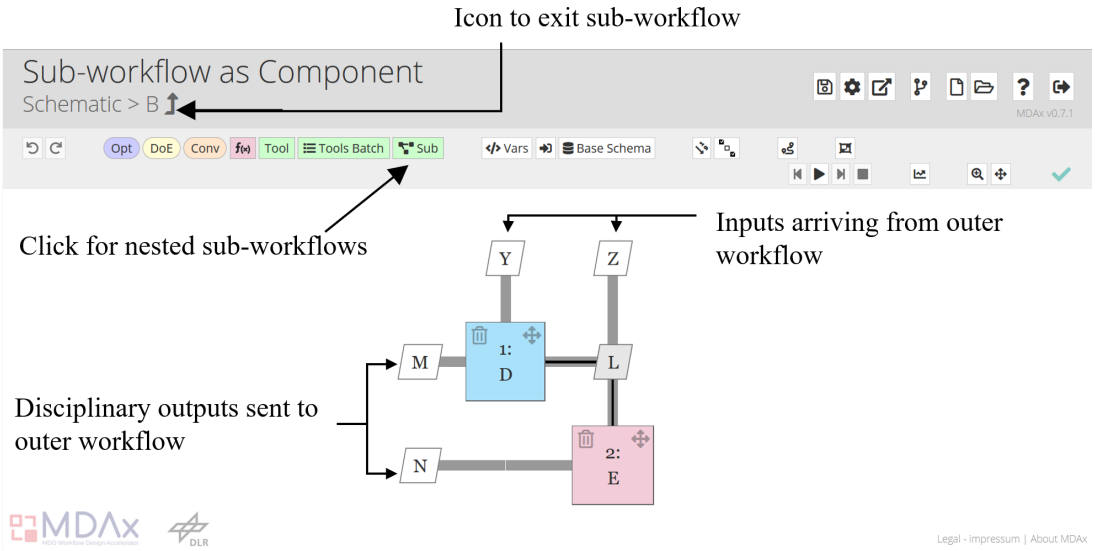
For the UI implementation of the "execution tool naming" feature, another field is added to the tool edit window, namely, 'Execution Name'. This is separate from the 'Tool Name' field which represents the name of the disciplinary black-box component itself. By default, the execution name is set as tool name. However, in the event that the user wants to use a multi-purpose back-end tool for more than one disciplinary component blocks, the user needs to provide the same back-end tool name as the execution name for all the disciplinary blocks. The export to RCE includes this name in the metadata, eliminating the need for creating a manual link in RCE. Figure 7 shows an example of a four engine aircraft with two different size engines on each wing, namely, inner and outer engine. However, since the tool for engine sizing remains the same, they both have the same execution name of the back-end tool as 'EngineSizer'.

Figure 8 shows the UI implementation of the sub-value matcher feature, using a conceptual example consisting of two tools, 'A' and 'B'. As can be seen in the variable tree of the CDS on the left, the tools are not specified using a dedicated name, but are instead following a specified schema where the parent node is 'tool' and the name of the tool is stored in the sub-node 'name'. However, since MDAx does not store any variable values, this presents a problem when the workflow is exported for execution since the execution platform can no longer differentiate the location from which the inputs and outputs have to read or written, respectively. This leads to addition work for the user where the location of the data needs to be hard coded at the RCE tool level to avoid confusion with the datasets for other tools. In such a case, a sub-value matcher can be specified for the differentiating sub-node 'name' and the specific name of the tool can be written there. This adds a special attribute to the parent node 'tool' which provides the indication to the execution platform on where to read and write the variables for tool A and tool B. Therefore, the user can identify where

[1] https://vuejs.org/



(a) Outer workflow with coupling of sub-workflow inputs and outputs with other disciplines



(b) Sub-workflow window similar to original window with possibility for multi-level workflow nesting

Figure 6 – UI for Sub-workflow as component in a XDSM with arrows to navigate between workflows

the variable nodes of a specific tool are located and how they are coupled with other tools, without confusion and in a time efficient manner. It also eliminates the risk of data being overwritten by other tools at the time of execution.

5. Practical Application Cases

The main objectives behind the creation of MDAx were laid down in [15]. These objectives were classified into three categories based on the possible use-cases: Exploration, Verification and Documentation. The exploration category involves setting up and inspecting available workflow components, organizing and utilizing a common data schema, and creating a workflow model. The verification category focuses on inspecting the selected workflow configuration, ensuring the proper conditions for workflow execution, and checking the connections among workflow components. Lastly, the documentation category addresses the formalization and communication of MDAO workflow models. This section will highlight some projects and their corresponding application cases where MDAx has played a significant role in past years. In a product design process, MDAx's role begins with helping setup the CDS and maintaining it, and ends with communicating an executable workflow to the PIDO platform. This means that the most effective use of MDAx in a project yields a viable application for

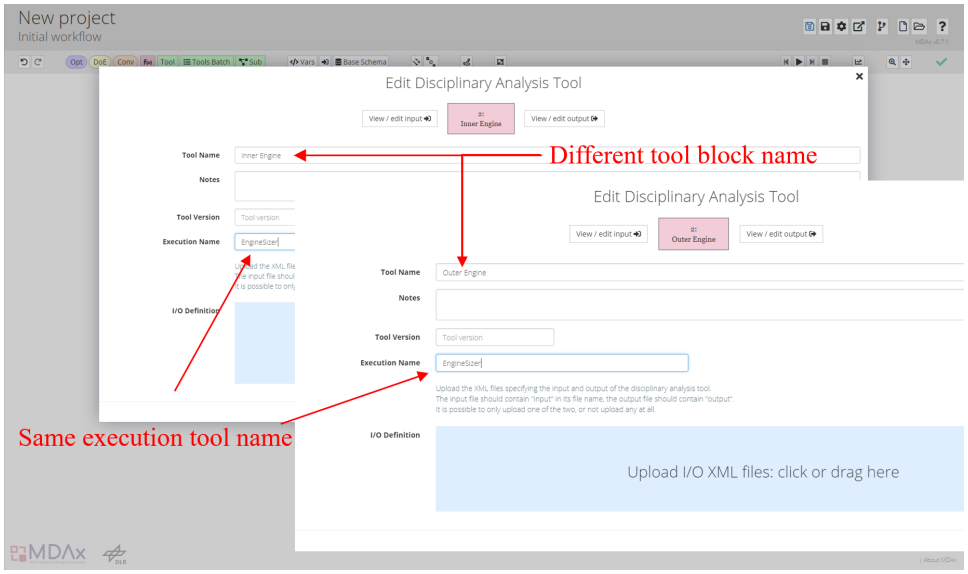


Figure 7 – UI for Execution Tool Naming with arrows showing separate tool name for the inner and outer engines which are using the same execution tool name, 'EngineSizer' at the back-end.

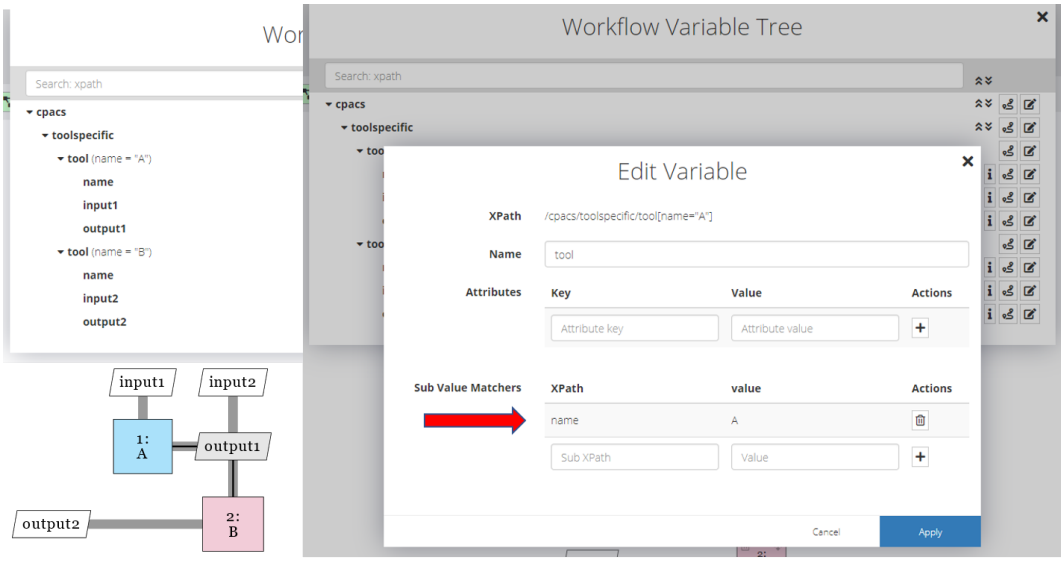


Figure 8 – UI for sub-value matcher feature where the differentiable sub-node and its value can be specified at the parent node to differentiate between the correct variable locations at the time of execution

not just one but all the three aforementioned use-case categories. Thus, it should be noted that it would not be ideal to segregate the application cases into these categories but to keep them in mind while discussing each application.

MDAx found it's first application in the project AGILE 4.0 [24]. The purpose of the project was to develop new methods and technologies to improve, streamline and accelerate the development of complex systems [25]. AGILE 4.0 focused on the aeronautical supply-chain: design, production, certification and manufacturing, with dedicated application cases for each. MDAx was developed as part of the AGILE 4.0 project and therefore, found a viable application in the design process aircraft configurations involved in the project. MDAx was used to identify the tool interfaces with CDS and the couplings between them to yield an executable workflow. The workflow was constantly updated based on any modifications to the tools or change in the aircraft configuration. An aspect of design which was addressed in this project was of aircraft family design. To that end, MDAx saw an expansion in the application of the software beyond single aircraft architecture design cases. However, the dynamic MDAO feature of MDAx was not developed until then and therefore, a workaround was found using

the "single static MDAO" approach, described in [22]. This ensured that all architecture choices are handled within the tool blocks themselves by the programmer. Figure 9 shows the XDSM generated for the overall aircraft design of the 3 members of the aircraft family where each block has 3 separate instances to address each of the three members. Each of these instances of the overall aircraft design (OAD) block is a workflow in itself which does the design analysis of a specific aircraft family member based on the inputs. It should be noted, that although the inputs and outputs of the three instances is the same, their values differ due to the aircraft having different top-level requirements. Further details, XDSMs and results can be found on the official website[2].

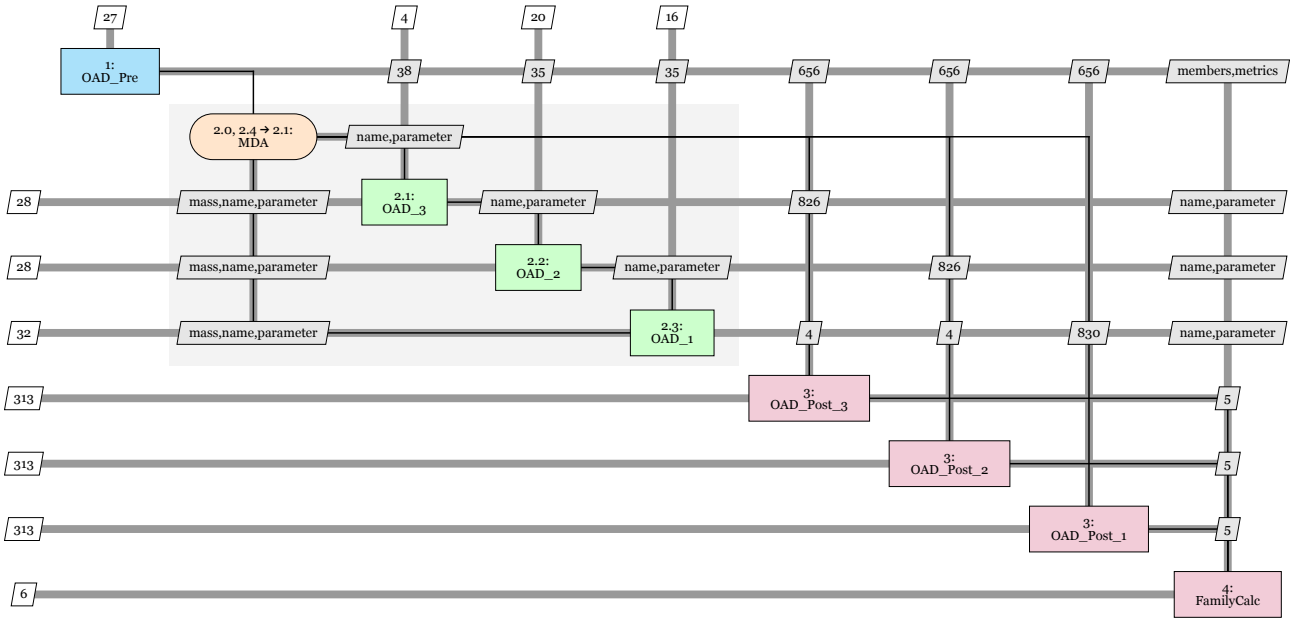


Figure 9 – XDSM for the design of a family of three business jets, trading-off aircraft performance and investment costs by varying the amount of shared components [26]

Another important application case was found as part of the DLR project "Exploration of Electric Aircraft Concepts and Technologies Projects" (EXACT) whose aim was to design aircraft concepts for climate-friendly flying while analysing and evaluating the entire life cycle of the aircraft [23]. Since the goal of the project was to investigate novel concepts, there remained a significant gap between the competences required to successfully analyse the configuration and the available competences among the partner organizations. Therefore, in order to identify and consolidate the available and the missing competences, MDAx was employed. Note, that over a 100 experts from over 30 different DLR institutes were part of the project. MDAx was used as an interface and platform for these experts to discuss the competences and define the necessary inputs that such competences would require in order to provide the desired outputs. This also helped the experts identify the possible coupling that such competences would have which in-turn provided the knowledge of how they would affect each other. Figure 10 shows an example of the competence overview created for the life cycle analysis part of the project. It can be seen that in the beginning, a larger number of competences were laid down. These were hypothetical and even included some competences which were out of the scope of the project. However, these were translated into real tools during the course of the project, which lead to an increase in the detailed knowledge about these competences. Thus, some inconsistencies and redundancies were removed, some competences were merged due to higher correlation and some were deemed out of scope. By the end of the project a more realistic XDSM was achieved (lower XDSM in Figure 10) where each of the competence had an analysis tool to carry out the task. A follow-up project of EXACT, called EXACT 2, has recently commenced, where MDAx is used to a wider extent to develop and run the overall aircraft design workflow and not just focus on a single aspect of the whole design chain. Currently, the 'subworkflow as component' feature has been employed to create the overall project workflow involving all the discipline. The next task would

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

to refine the definition of the inputs and outputs of the various tools and generate an executable workflow. It is planned to employ the new "Dynamic MDAO" feature of MDAx, if possible, at a later stage in the project for the 'Drive and System Design' discipline. An insight into the latest version of the overall workflow can be see in Figure 11

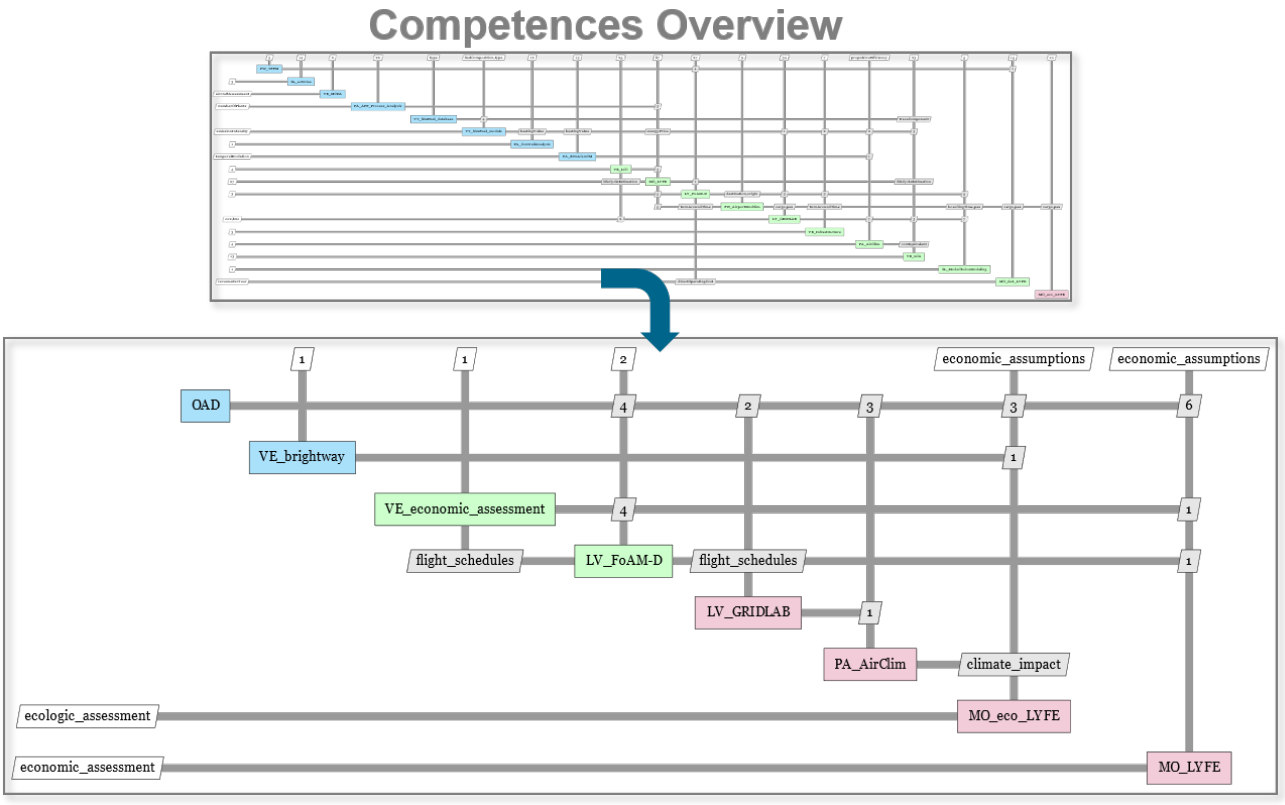


Figure 10 – An overview of the initially identified competences for life cycle assessment of aircraft as part of EXACT project along with the finalised workflow at the end of the project based on developed competences.

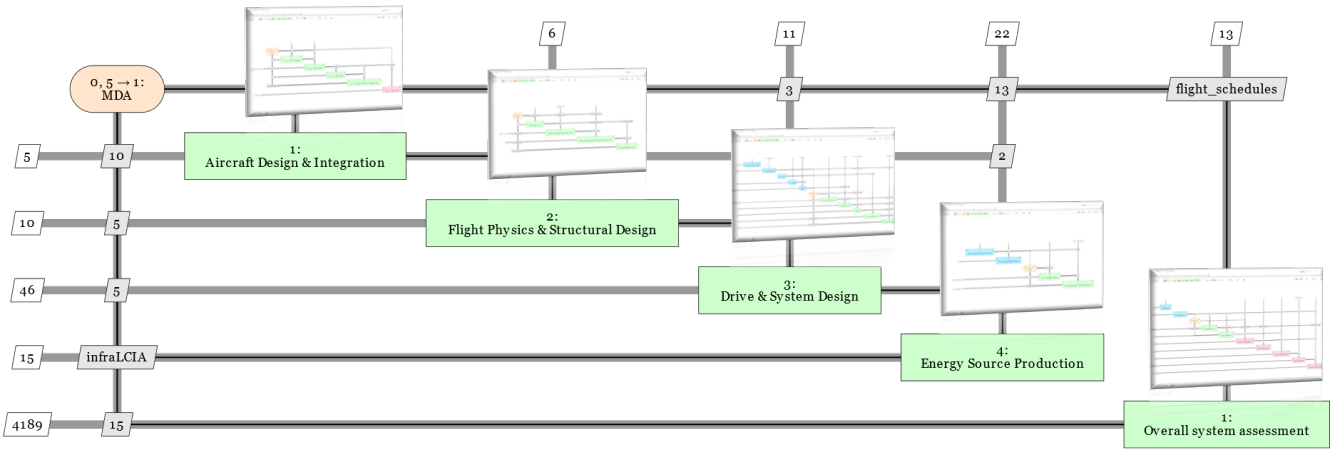


Figure 11 – An XDSM overview of the different aspects of aircraft design to be addressed as part of the EXACT 2 project where each disciplinary block is a sub-workflow in itself.

EXACT and EXACT 2 are DLR internal project involving experts across DLR institutes. The use of MDAx in these projects has helped establish a base for using digital methods for collaborative MDAO within DLR. Another project which has helped in this task is ALICIA (Aviation Life Cycle and

Impact Assessment) where the aim is to develop a digital platform for modeling and assessing aircraft over the entire life-cycle [3]. MDAx was again used as a platform to bring together all the experts and their respective tools together in order to help automatically identify the couplings between the respective tools and refine the CDS according to nomenclatures acceptable to all the partners. The resultant workflow for the air transport system (ATS) level analysis can be seem in Figure 14 in Appendix C. Project ProTekT (Probabilistic Technology assessment of complex Transport systems) is another initiative where MDAx has played an instrumental role [27]. The project focuses on developing methods for the technological assessment of complex system architectures that take uncertainties into account. A project XDSM can be seen in Figure 15 in Appendix D.

Furthermore, just like the AGILE 4.0 project, there are an increasing number of international projects where MDAx is being used. One such project is Impact Monitor where the goal is to develop a comprehensive and integrated impact assessment framework and toolbox that aids the European Commission (EC) in making informed, science-based decisions regarding the environmental, economic, and societal impacts of European aviation research and innovation (R&I) policies and technologies[4]. An insight into the workflow generated using MDAx for the continuous decent operations application case can be seen in Figure 16 in Appendix E. Apart from these, there have been initiative where the collaborative engineering framework of DLR, which consists of CPACS, MDAx and RCE, has been disseminated and introduced to international agencies across the work. A good example of such an initiative was the IFAR-ECN event which was mentioned earlier in the paper [8]. Young experts from 18 countries across 4 continents were trained on the framework with the hope of establishing a standard for approaching multi-disciplinary design tasks, across the world. The workflow can be seen in Figure 12 in Appendix A. Lastly, initiatives by Korea Aerospace Research Institute (KARI) in the field of MDAO has already made it evident that MDAx is finding it's reach internationally [28]. An example workflow generated using MDAx for their hybrid eVTOL aircraft application case can be seen in Figure 17 in Appendix F.

6. Conclusion and Outlook

Conventionally, simulation workflows are constructed manually, involving time-consuming and repetitive tasks. Modifying these workflows, such as adding or removing simulation tools or enhancing tool fidelity, demanded considerable effort, particularly in complex systems with numerous interacting disciplines. MDAx addresses these challenges by offering a flexible modeling environment for simulation workflows. Its XDSM-centric design simplifies workflow modeling, allowing engineers without extensive MDAO knowledge to engage in and lead collaborative design studies. With all workflow components treated as black boxes, MDAx is versatile and applicable across a wide range of engineering disciplines.

The current paper reiterates the theory and methodology behind the development of MDAx, while highlighting some of the latest additions to the features provided by the software. One of these is the possibility of specifying a "subworkflow as component" in the MDAO workflows. It allows disciplinary experts to save and reuse their competence tools chains as a black-box in a bigger workflow. Thus, enabling nested workflows for more efficient integration and updates. The corresponding UI for the same has also been implemented in the software. Another important addition was to enable the creation of "Dynamic MDAO" workflows. Dynamic MDAO is particularly valuable in System Architecture Optimization (SAO) problems which explore optimal architectures within a large combinatorial design space, using MDAO to evaluate different architectures considering all coupled design disciplines. In order to achieve this, a collaborative MDAO platform is required that can automatically adjust the MDAO problem for each architecture. Out of the various strategies for applying MDAO in SAO, a suitable one for dynamic modifications is implemented in the MDAx platform. This involves defining architectural influences that guide necessary modifications in the MDAO formulation. Some other features were also implemented like allowing multiple disciplinary blocks to have a common back-end tool for execution and also, enabling sub-value matching which allows for a value based variable identification instead of name based.

[3] https://event.dlr.de/en/ila2024/alicia/

[4] https://impactmonitor.eu/

A few application cases, where MDAx has played a role in establishing the MDAO workflow, are mentioned in this paper. These applications not only include internal project across many DLR institutes, but also include European and other international project where partners from all over the world have worked on MDAx. This is proof enough of the fact that such MDAO platforms are needed for a faster and more efficient design process. Moreover, some of these examples showed the application of the recently added features already. This shows the fact the current development of the software is synonymous with the expert requirements.

The next step in further development of MDAx will be to implement the UI for the dynamic MDAO feature. This would also need a modification to the description of XDSMs to include the dynamic MDAO features. Moreover, it can be foreseen that implementing dynamic MDAO features for sub-workflows might pose a challenge that needs to be overcome. One way achieve this, would be to apply the dynamic MDAO feature to groups of tools rather than individual tools. Another step, is to improve MDAx for higher fidelity collaborative MDAO applications. This would include enabling methods to specify surrogate models and reduced order models. This would further lead to a need for specifying parameters for uncertainty quantification of the uncertainties that arise due to surrogates and approximations. Furthermore, MDAx has a well established connection with the workflow execution phase which follows the workflow formulation. This can be made even stronger by implementing the export of novel feature like sub-workflow as component and dynamic MDAO for other MDAO platforms and PIDO platforms apart from RCE. On the other hand, the connection of MDAx with the system architecting phase which precedes it, is still manual and requires methodology development to enable automation.

7. Contact Author Email Address

E-mail: Sparsh.Garg@dlr.de
Phone: +49 402 489641329
Address: Hein-Saß-Weg 22, 21129 Hamburg, Germany

8. Copyright Statement

Appendix A

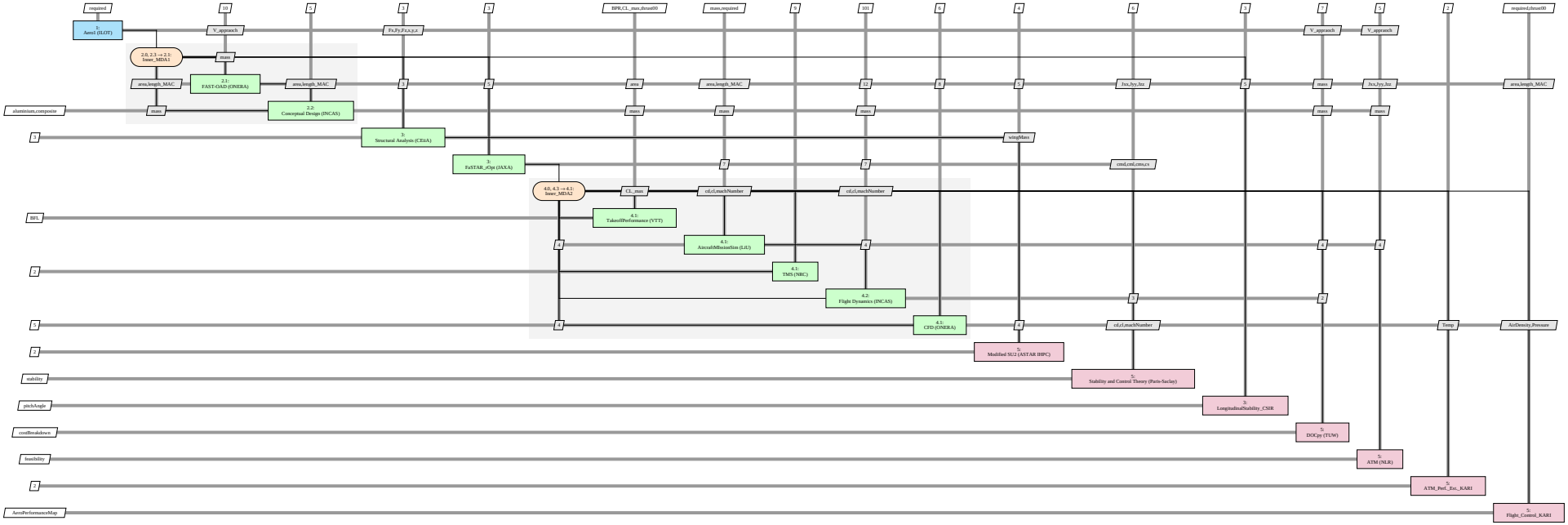


Figure 12 – XDSM with 15 competences addressing various aircraft design disciplines within the MDAO workflow model for design of a sustainable aircraft as part of the IFAR-ECN 2023 Conference [8]

Appendix B

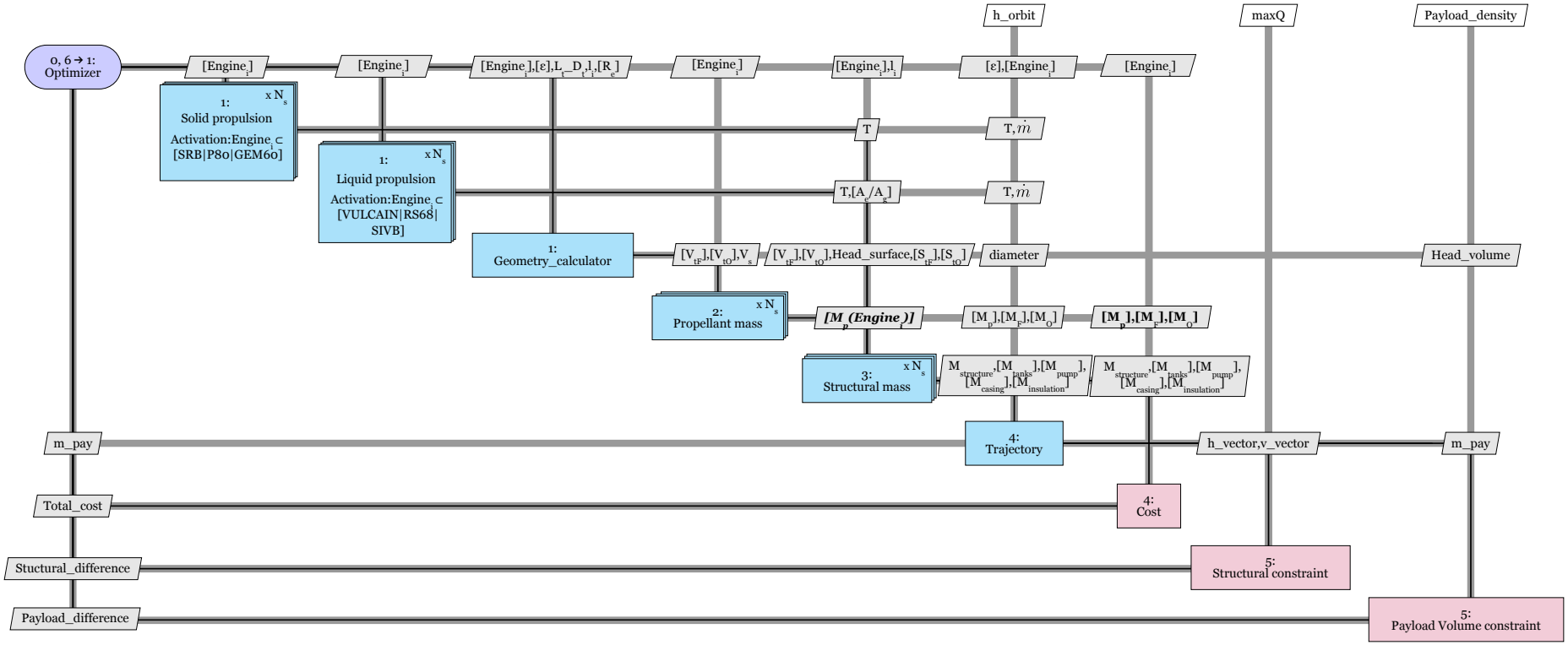


Figure 13 – XDSM of the space rocket benchmark problem depicting the analysis disciplines, constraints and the variable couplings between them. Dynamic XDSM is a provisional extension of XDSM to include the four architectural influences: *Conditional variables* are indicated within square brackets. Variables involved in *data connection* are expressed in bold and italic, including between parenthesis the architectural decision causing them. *Discipline repetition* is indicated in the top right of the discipline block where the multiplicity is a result of an architectural decision. Lastly, the activation condition behind *discipline activation* is expressed in an additional row in the discipline block.

Appendix C

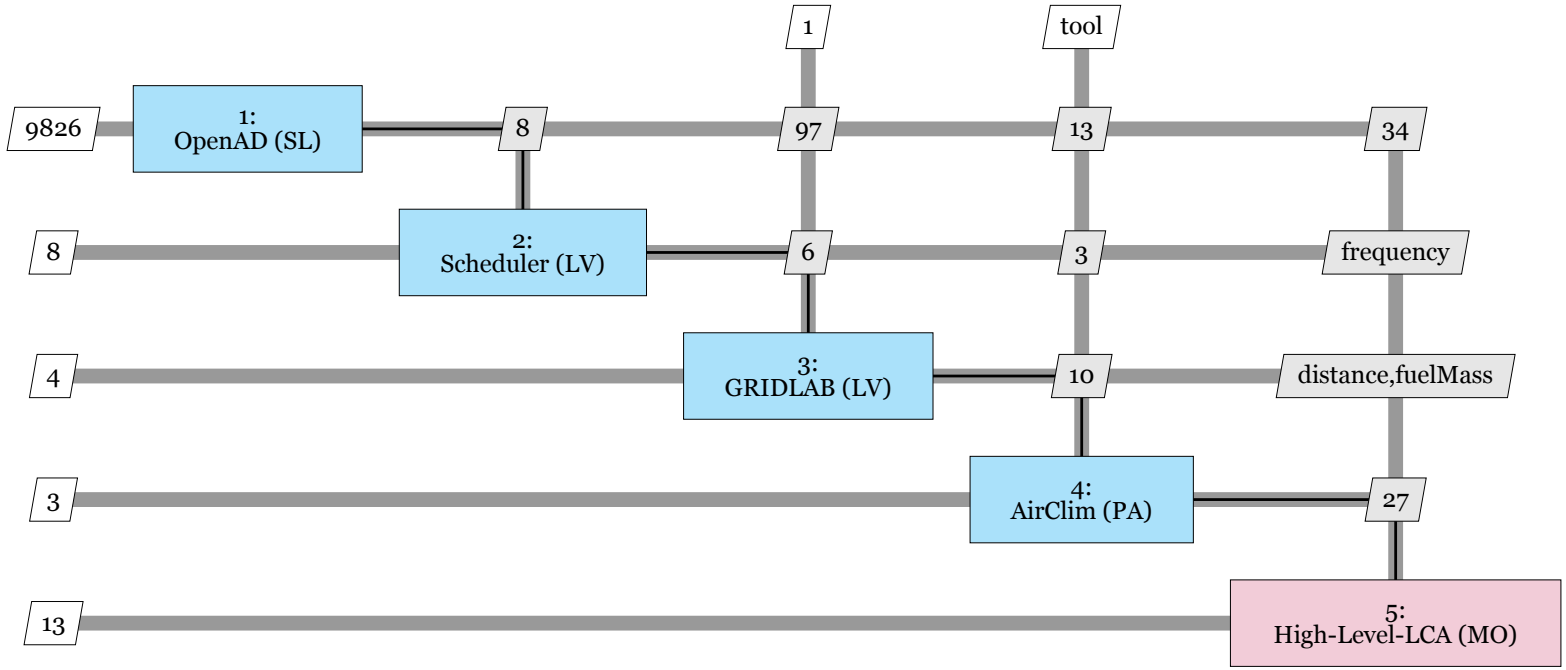


Figure 14 – An XDSM overview of the air transport system (ATS) level analysis of the aircraft including climate and life cycle analysis disciplines as part of ALICIA project

Appendix D

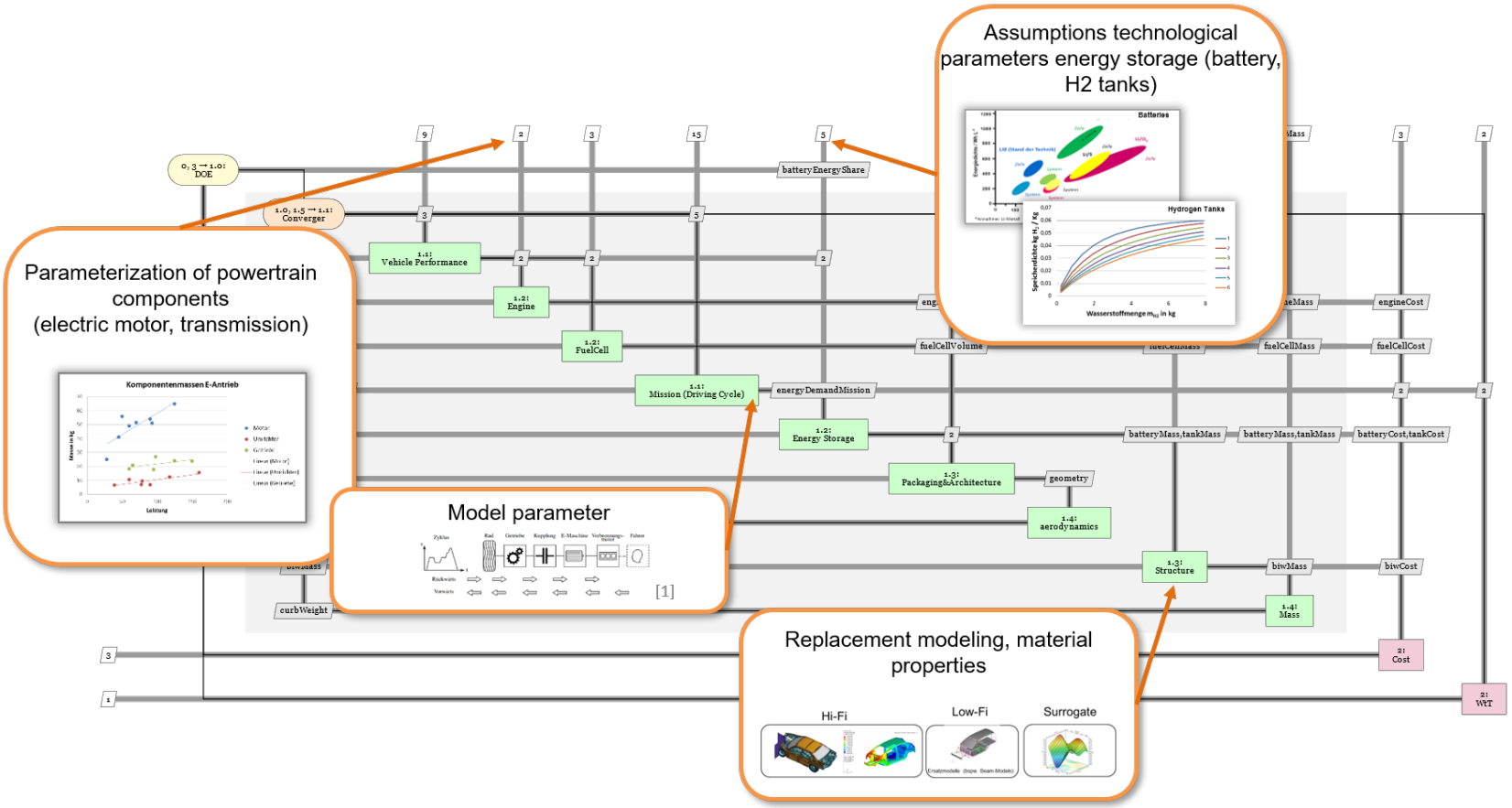


Figure 15 – XDSM generated as part of the project ProTekT for technological assessment of transport systems

Appendix E

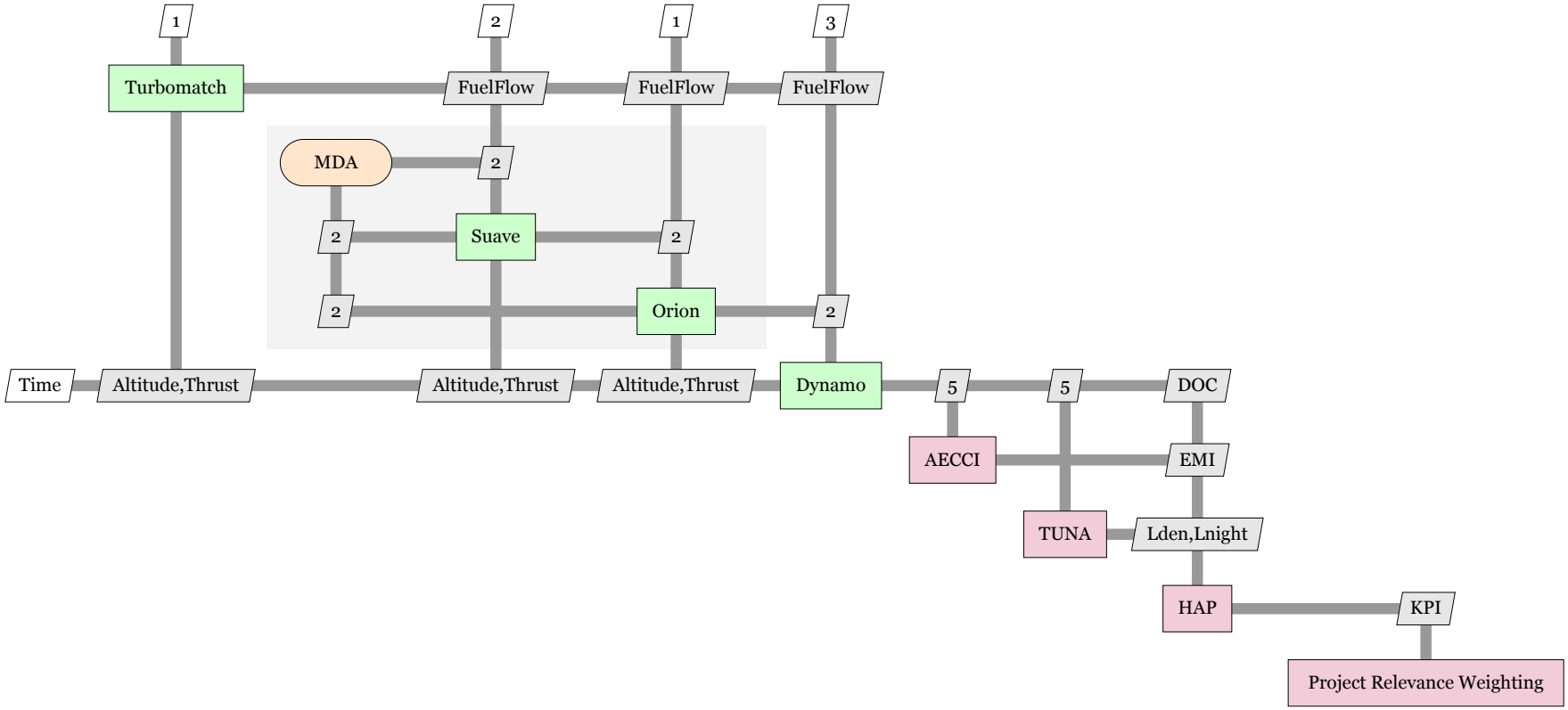


Figure 16 – A preliminary XDSM generated for the continuous descend operation application case of the Impact Monitor Project

Appendix F

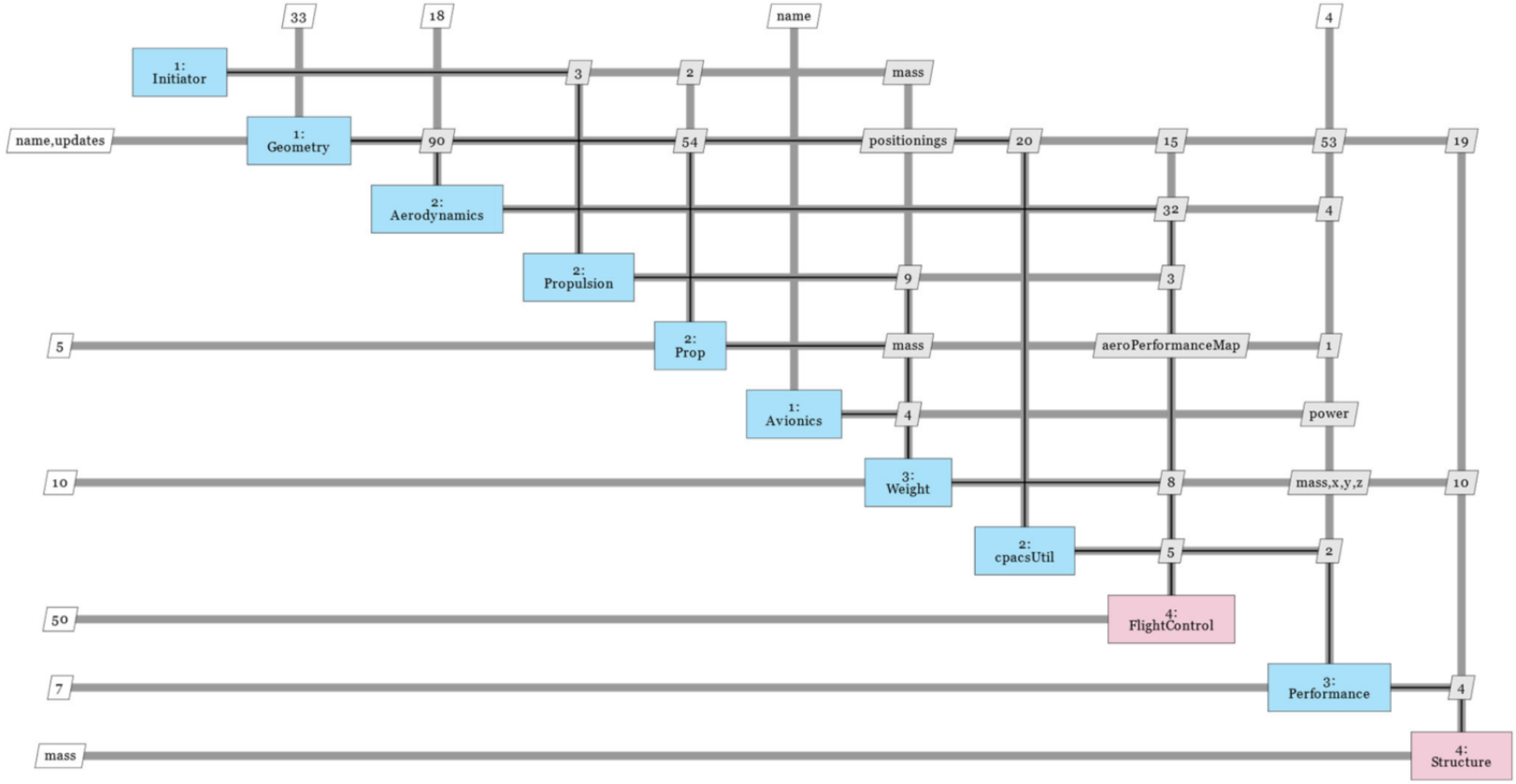


Figure 17 – Use of MDAx by KARI for the MDAO analysis of their hybrid eVTOL aircraft application case

References

[1] Pier Davide Ciampa and Björn Nagel. Accelerating the development of complex systems in aeronautics via mbse and mdao: a roadmap to agility. In *AIAA AVIATION 2021 FORUM*, Reston, Virginia, 2021. American Institute of Aeronautics and Astronautics.

[2] Jasper Bussemaker, Luca Boggero, and Björn Nagel. The agile 4.0 project: Mbse to support cyber–physical collaborative aircraft development. *INCOSE International Symposium*, 33(1):163–182, 2023.

[3] Jaroslaw Sobieszczanski-Sobieski, Alan Morris, and Michel Van Tooren. *Multidisciplinary design optimization supported by knowledge based engineering*. John Wiley & Sons, 2015.

[4] Joaquim R. R. A. Martins and Andrew Ning. *Engineering design optimization*. Cambridge University Press, Cambridge, 2022.

[5] Joaquim R. R. A. Martins and Andrew B. Lambe. Multidisciplinary design optimization: A survey of architectures. *AIAA Journal*, 51(9):2049–2075, 2013.

[6] Pier Davide Ciampa and Björn Nagel. Agile paradigm: The next generation collaborative mdo for the development of aeronautical systems. *Progress in Aerospace Sciences*, 119:100643, 2020.

[7] Pier Davide Ciampa and Björn Nagel. Towards the 3rd generation mdo collaborative environment. *30th ICAS*, pages 1–12, 2016.

[8] Sparsh Garg, Ana M. P. Silva, Pedro F. Albuquerque, Eric Nguyen Van, Keita Kimura, Yosuke Sugioka, Marcos de R. Jacinto, and Prajwal Shiva Prakasha. IFAR-X: Collaborative engineering framework for next generation of aerospace engineers working on aircraft design. *34th ICAS Congress*, 2024, inproceedings.

[9] Erik Baalbergen, Jos Vankan, Luca Boggero, Jasper H. Bussemaker, Thierry Lefebvre, Bastiaan Beijer, Anne-Liza Bruggeman, and Massimo Mandorino. Advancing cross-organizational collaboration in aircraft development. In *AIAA AVIATION 2022 Forum*, Reston, Virginia, 2022. American Institute of Aeronautics and Astronautics.

[10] Marko Alder, Erwin Moerland, Jonas Jepsen, and Björn Nagel. Recent advances in establishing a common language for aircraft design with cpacs. 01 2020.

[11] Justin S. Gray, John T. Hwang, Joaquim R. R. A. Martins, Kenneth T. Moore, and Bret A. Naylor. Openmdao: an open-source framework for multidisciplinary design, analysis, and optimization. *Structural and Multidisciplinary Optimization*, 59(4):1075–1104, 2019.

[12] Imco van Gent, Gianfranco La Rocca, and Leo L. Veldhuis. Composing mdao symphonies: graph-based generation and manipulation of large multidisciplinary systems. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, AIAA AVIATION Forum, Reston, Virginia, 2017. American Institute of Aeronautics and Astronautics.

[13] François Gallard, Charlie Vanaret, Damien Guénot, Vincent Gachelin, Rémi Lafage, Benoît Pauwels, Pierre-Jean Barjhoux, and Anne Gazaix. Gems: A python library for automation of multidisciplinary design optimization process generation. 01 2018.

[14] Imco van Gent, Benedikt Aigner, Bastiaan Beijer, and Gianfranco La Rocca. A critical look at design automation solutions for collaborative mdo in the agile paradigm. In *2018 Multidisciplinary Analysis and Optimization Conference*, Reston, Virginia, 06252018. American Institute of Aeronautics and Astronautics.

[15] Andreas Page Risueño, Jasper Bussemaker, Pier Davide Ciampa, and Bjoern Nagel. Mdax: Agile generation of collaborative mdao workflows for complex systems. In *AIAA Aviation 2020 Forum*, page 3133, 2020.

[16] I. van Gent. Agile mdao systems: A graph-based methodology to enhance collaborative multidisciplinary design. 2019.

[17] Andrew B. Lambe and Joaquim R. R. A. Martins. Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. *Structural and Multidisciplinary Optimization*, 46(2):273–284, 2012.

[18] Brigitte Boden, Jan Flink, Niklas Först, Robert Mischke, Kathrin Schaffert, Alexander Weinert, Annika Wohlan, and Andreas Schreiber. Rce: An integration environment for engineering and science. *SoftwareX*, 15:100759, 2021.

[19] Edward Crawley, Bruce Cameron, and Daniel Selva. *System Architecture: Strategy and Product Development for Complex Systems*. 04 2015.

[20] J. H. Bussemaker and Pier Davide Ciampa. *MBSE in Architecture Design Space Exploration*, pages 589–629. Springer International Publishing, 2023.

[21] J. H. Bussemaker, L. Boggero, and B. Nagel. System architecture design space exploration: Integration with computational environments and efficient optimization. In *AIAA AVIATION 2024 FORUM*, Las Vegas, NV, USA, July 2024.

[22] Sparsh Garg, Raul García Sanchez, Jasper H. Bussemaker, Luca Boggero, and Björn Nagel. Dynamic formulation and execution of mdao workflows for architecture optimization. *AIAA Aviation*, 2024, inproceedings.

[23] Hartmann, J., Nagel, B.: Eliminating climate impact from aviation - a system level approach as applied in the framework of the dlr-internal project exact (exploration of electric aircraft concepts and technologies). German Aerospace Congress (2021).

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

[25] *THE APPLICATION OF THE AGILE 4.0 MBSE ARCHITECTURAL FRAMEWORK FOR THE MODELING OF SYSTEM STAKEHOLDERS, NEEDS AND REQUIREMENTS*. Zenodo, November 2021.

[26] Jasper Bussemaker, Pier Davide Ciampa, Jasveer Singh, Marco Fioriti, Carlos Cabaleiro, Zhijun Wang, Daniël Peeters, Philipp Hansmann, Pierluigi Vecchia, and Massimo Mandorino. Collaborative design of a business jet family using the agile 4.0 mbse environment. 06 2022.

[27] Marko Alder, Tawfiq Ahmed, Benjamin Fröhler, and Anna Skopnik. Assessment of techniques for global sensitivity analyses in conceptual aircraft design. 06 2023.

[28] Seung-Kil Paek. Kari-dlr digital collaboration to build a mdao system for the preliminary design of hybrid evtol aircraft. *33th ICAS Congress*, 2022.