DOI: ADD DOINUMBER HERE

Dynamic workflow generation applied to aircraft moveable architecture optimization

J.S. Sonneveld[*][†], *T. van den Berg*[**], *G. la Rocca*[*], *S. Valencia-Ibáñez*[*], *B. van Manen*[**]
A.M.R.M. Bruggeman[*], *B. Beijer*[***]

[*] Delft University of Technology, Delft, 2629 HS, The Netherlands
[**] GKN Fokker, Papendrecht, 3351 LB, The Netherlands
[***] KE-chain, Delft, 2629 JD, The Netherlands

J.S.Sonneveld@tudelft.nl – Tobie.vandenberg@fokker.com – G.LaRocca@tudelft.nl –
S.ValenciaIbanez@student.tudelft.nl – Bas.vanmanen@fokker.com –
A.M.R.M.Bruggeman@tudelft.nl – Bastiaan.Beijer@ke-chain.com
[†] Corresponding Author

Abstract

This paper discusses the approach for architecture design space optimization of aeronautical systems investigated in the DEFAINE project [1]. In system architecture optimization problems, hierarchical relations between design variables may exist. This means that the quantity and type of some variables are dependent on the value of other variables. To address this challenge, a nested optimization strategy is proposed, where an outer loop deals with the independent design variables and an inner loop with the dependent ones. As the characteristics of the dependent variables may become known only during workflow execution, a dynamic workflow formulation approach is developed to introduce these variables into the MDAO workflow, automatically, during execution. The proposed methodology is implemented using a suite of technologies provided by partners of the DEFAINE consortium to enable formulation and execution of collaborative MDAO problems. This implementation is then applied to the design and optimization of the structural layout of a UAV aileron. The goal is to minimize the aileron mass, subjected to failure criteria constraints. In this use case, hierarchical relations between variables are present. The aileron skin is discretised into material zones, each one defined by a (dependent) design variable. The number of zones, hence the number of design variables, is not known a priori as it depends on the number and position of ribs and spars (both independent variables). A preliminary implementation of the proposed approach proved effective in dealing with the hierarchical mixed-integer design space. Trade off studies comparing different aileron architectures could be efficiently set up and executed. Some limitations in the implemented workflow management technology, however prevent performing a full nested architecture optimization. Therefore, to demonstrate the effectiveness of the proposed nested approach, a parallel study was conducted on the same aileron, using a less generic and manual implementation of the nested MDAO workflow. This second study successfully minimized the mass and cost of the aileron structure, thereby proving the merit of the proposed nested approach to address system architecture optimization.

Nomenclature

CMDOWS	=	Common MDO Workflow Schema
DOE	=	Design Of Experiments
DSC	=	Design Study Configuration
KADMOS	=	Knowledge- and graph-based Agile Design for MDO Systems
KBE	=	Knowledge Based Engineering
MDAO	=	Multidisciplinary Design Analysis and Optimization
OEM	=	Original Equipment Manufacturer
PIDO	=	Process Integration and Design Optimization
QOI	=	Quantity of Interest
RCE	=	Remote Component Environment
RF	=	Reserve Factor
UAV	=	Unmanned Aerial Vehicle
XDSM	=	eXtended Design Structure Matrix
XML	=	eXstensible Markup Language

1. Introduction

Consolidated strategies and optimization algorithms can be used to perform multidisciplinary and multi-objective optimization of engineering products, as far as the architecture of the system to be optimized is fixed. Accounting for the complete system architecture design space in an optimization process is very challenging because of the presence of integer and categorical design variables which lead to a combinatorial explosion of designs to be evaluated. An extra challenge arise because of the hierarchical relationships that may exist between design variables, i.e., the number and/or existence of certain design variables may depend on the value assumed by other design variables. In fact, the hierarchical nature is generally a consequence of the aforementioned categorical variables, as the presence of certain components in the system architecture comes with the necessary design variables to define said components. This makes the design vector of the optimization problem dynamic and unknown a priori.

A practical approach to handle such hierarchical problems is to split them into multiple but smaller and more tractable sub-problems, hence focusing on (the optimization of) multiple preselected architectures. However, this approach prevents a true exploration of the design space and suffers from the initial architecture selection biases. Bussemaker et al. [2] propose an optimization strategy using a fixed length design vector including all possible design variables. A so-called imputation strategy is implemented to inform the optimizer to deactivate, at each iteration, the design variables made irrelevant by the value assigned to other variables, higher in the hierarchy. This approach is not trivial and may result in poor computational efficiency. Additionally, since the complete design vector must be formulated a priori, a complete model of the product architecture hierarchy is required. However, this is not always possible a priori, as the effect of certain design variables on number and type of other design variables can only be established during simulation.

An alternative approach is to pursue a nested strategy, where an outer loop manages the high-level architecture variables, while an inner loop is dynamically formulated, based on the value of the outer loop variables, and executed. This, in turn, can lead to other dynamically formulated nested loops. Frank et al. [8] employed a nested strategy to simultaneously compare and optimize architectures and successfully applied it to suborbital vehicle design. Michalek and Papalambros [9] combined the analytical target cascading multidisciplinary optimization architecture with a branch-and-bound outer loop to handle discrete variables. The formulation and execution of these nested optimization studies are generally problem specific and rely on ad hoc solutions and expensive manual implementation. Besides, the problem related to the unknown number of some design variable, as depending on the value assumed by other design variables stands.

This paper proposes a methodology and two technical implementations, to address the aforementioned challenges, using a nested approach, based on the dynamic formulation of the inner loop workflow. The methodology and its main components are described in Section 2. Section 3 describes a technical implementation based on technologies provide by the DEFAINE project partners to formulate and execute MDAO workflows in a collaborative environment. Section 4 discusses the application of the proposed implementation to the architecture design space exploration of UAV aileron. In the same section, a second implementation of the proposed methodology is discussed, which enabled actual multidisciplinary and multi-objective optimization of the same aileron. Conclusions and outlook are given in Section 5.

2. Methodology

This section introduces a methodology developed for setting-up and executing system architecture optimization problems using a nested approach, based on dynamic formulation of the inner loop workflow(s). In Section 2.1, a format for configuring nested design space exploration and optimization studies is proposed. In Section 2.2, the process to translate a configured design study into a series of nested workflows is explained. The dynamic workflow reformulation approach to handle introducing dependent variables into the workflow formulation is introduced in section 2.3.

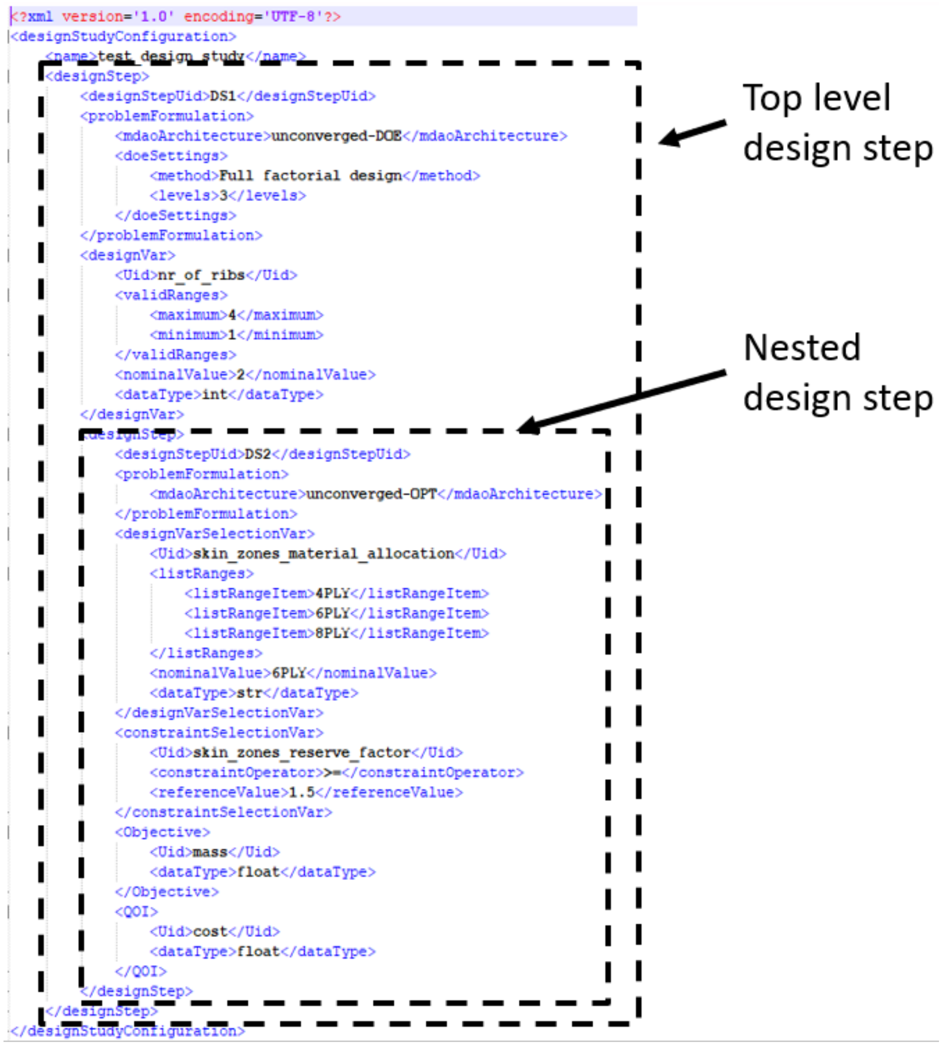
2.1 Design Study Configuration

As a means to configure a hierarchical design study, the eXtensible Markup Language (XML)-based Design Study Configuration (DSC) schema was developed. In a DSC file, the hierarchical variable structure of a product can be configured, allowing for unknown variable quantities and types.

A DSC file features a number of nested design steps. The variables specified in a nested design step depend on the variables configured in higher level design steps. In this way, the hierarchy of the problem is configured. In principle there is no limit on the amount of nested design steps that can be specified. Within each design step it is mandatory to specify a '**designStepUid**' and '**problemFormulation'**. The problem formulation governs what type of MDAO strategy will be used in the design step, i.e., a Design of Experiments (DOE) or an actual multidisciplinary optimization architecture. For each design step, one or more of the following nodes can be specified in the DSC schema:

- **designVar:** design variable, including the bounds if the variable is continuous/integer, or a list of options if the variable is categorical.
- **designVarSelectionVar:** design variables whose type and quantity are not known a priori, but depend on the value of design variables in a higher design step. Cannot be specified in a top-level design step.
- **Constraint:** constraint variable, including the constraint operator and reference value
- **ConstraintSelectionVar:** a constraint, of which the type and quantity is not known a priori and depends on the value of design variables in a higher design step. Cannot be specified in a top-level design step.
- **Objective:** the quantity of interest, or performance indicator of the system to be improved
- **ObjectiveSelectionVar:** an objective whose quantity is not known a priori and depends on the value of design variables in a higher design step. Cannot be specified in a top-level design step. Having a parameter like this defined, means that the optimization might become multi-objective if the quantity turns out to be more than 1.
- **QOI:** quantity of interest
- **QOISelectionVar:** a quantity of interest whose type and quantity is not known a priori and depends on the value of design variables in a higher design step. Cannot be specified in a top-level design step.
- **designStep:** Within each design step it is possible to specify one nested design step.

To clarify the structure of a DSC file and its nodes, we can consider the example of an aileron structural optimization. The goal is to minimize the aileron mass, while complying with failure constraints, by varying the number of ribs. The aileron skin is discretized into zones, where the amount of skin zones depends on the number of ribs. For each skin zone, a design variable is allocated to control the material thickness. A reserve factor can be calculated to check the material allowable loads in the skin zones are not exceeded. The DSC file describing this design study is shown in Figure 1. In the top-level design step, a DOE is configured with the number of ribs as design variable. Since the quantity of the 'skin zones material allocation' and 'skin zones reserve factor' variables are dependent on the number of ribs, they are respectively configured as '**designVarSelectionVar**' and '**constraintSelectionVar**' in the nested design step. The nested design step is configured as an optimization featuring mass as the objective and cost as a quantity of interest.



```xml
<?xml version='1.0' encoding='UTF-8'?>
<designStudyConfiguration>
    <name>test design study</name>
    <designStep>
        <designStepUid>DS1</designStepUid>
        <problemFormulation>
            <mdaoArchitecture>unconverged-DOE</mdaoArchitecture>
            <doeSettings>
                <method>Full factorial design</method>
                <levels>3</levels>
            </doeSettings>
        </problemFormulation>
        <designVar>
            <Uid>nr_of_ribs</Uid>
            <validRanges>
                <maximum>4</maximum>
                <minimum>1</minimum>
            </validRanges>
            <nominalValue>2</nominalValue>
            <dataType>int</dataType>
        </designVar>
        <designStep>
            <designStepUid>DS2</designStepUid>
            <problemFormulation>
                <mdaoArchitecture>unconverged-OPT</mdaoArchitecture>
            </problemFormulation>
            <designVarSelectionVar>
                <Uid>skin_zones_material_allocation</Uid>
                <listRanges>
                    <listRangeItem>4PLY</listRangeItem>
                    <listRangeItem>6PLY</listRangeItem>
                    <listRangeItem>8PLY</listRangeItem>
                </listRanges>
                <nominalValue>6PLY</nominalValue>
                <dataType>str</dataType>
            </designVarSelectionVar>
            <constraintSelectionVar>
                <Uid>skin_zones_reserve_factor</Uid>
                <constraintOperator>>=</constraintOperator>
                <referenceValue>1.5</referenceValue>
            </constraintSelectionVar>
            <Objective>
                <Uid>mass</Uid>
                <dataType>float</dataType>
            </Objective>
            <QOI>
                <Uid>cost</Uid>
                <dataType>float</dataType>
            </QOI>
        </designStep>
    </designStep>
</designStudyConfiguration>
```

Figure 1: Example of Design Study Configuration file instance featuring a top-level and nested design step

After constructing a DSC file, nested workflows must be formulated to evaluate the defined variables, this is explained in the next section. A dynamic workflow reformulation action will take care of introducing the '**–SelectionVar**' variables into the nested workflow formulations once the required information has become available. This principle is explained in section 2.3.

2.2 Nested Workflows Formulation

Once a design study is configured by constructing a DSC file as described above, for each configured design step an MDAO workflow must be formulated capable of evaluating all the specified variables. The workflow consists of one or more modelling and analysis tools, and in case of a nested design step, a tool that handles formulating and executing the MDAO workflow representing the nested design step. As an example, suppose a single multidisciplinary modelling and analysis tool handles all specified input and output variables. This tool is server-based such that 'set' and 'get' queries can be sent to impose inputs and retrieve outputs. For this setup the workflows can be constructed from the following standard tools:

- **Initialize-tool:** A multidisciplinary tool instance is initialized on a server, this tool instance is kept alive during the execution of the workflow, such that 'set' and 'get' queries can be send to it.

- **Set-get-tool:** inputs and outputs are imposed and requested using ‘set’ and ‘get’ queries to the previously initialized server-based tool instance. Among others, this includes the possibility to request information on the quantity and type of the dependent selection variables.
- **Next-design-step:** This tool takes care of formulating and executing a sub-workflow representing a nested design step, based on information retrieved using the set-get-tool block.
- **Post-processor:** An optional post processing step can take place; an example is uploading the results of an iteration to a (web-based) database.

The (X)DSM representations of a main workflow and nested sub-workflow formulated using the setup described above, for the two-step design study configured in Figure 1 are shown in Figure 2. The main workflow features the ‘initialize-tool’ block after which the DOE loop is started. Within the DOE loop, a ‘set-get-tool’ block handles all the input and output requests to the server-based tool. The next-design-step block takes care of formulating and triggering a nested sub-workflow. Within the sub-workflow, again the ‘set-get-tool’ is used to ‘set’ the design variables and ‘get’ the constraint/objective/QOI variables. Before any workflow execution has happened, the nested sub-workflow is still an incomplete workflow formulation as the quantity and type of the configured selection variables (introduced in Section 2.1) are not yet known. The selection variables placed in the sub-workflow are placeholders to be updated once their quantity and type has become known in a higher-level workflow. The dynamic aspect of handling the selection variables by reformulating the sub-workflow at each iteration is explained in the next section.

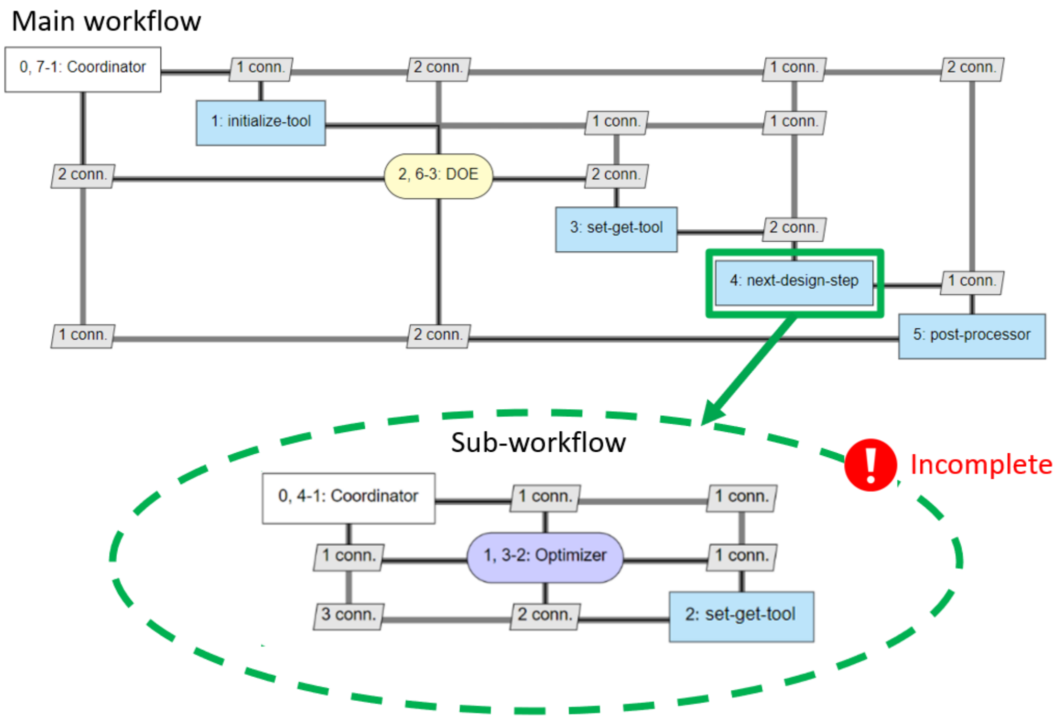


Figure 2: (X)DSM representations of a DOE main workflow featuring a nested optimization workflow, the nested workflow is still incomplete as the quantity and nature of the selection variables are not yet known.

2.3 Dynamic Workflow Reformulation

In Figure 3, an example of how selection variables are dealt within the ‘set-get-tool’ and ‘next-design-step’ blocks is illustrated. In this case, a ‘set’ and ‘get’ request are communicated to the ‘set-get-tool’ block. An independent design variable (nr_of_ribs) must be set, and two selection variables (skin_zones_material_allocation and skin_zones_reserve_factor) must be retrieved from the tool. After setting the number of ribs, the quantity and type of the selection variables are extracted and returned. In this case the number of ribs is set to two, this leads to three material zones and three reserve factors. Using this information, within the ‘next-design-step’ block the partially formulated sub-workflow is completed by adding the three material zones design variables, while the three reserve factors are set as three constraints. Looking at the sub-workflow in Figure 2 and Figure 3, it can be seen that the 1 input connection to ‘set-get-tool’ in the sub-workflow (the skin_zone_material_allocation selection variable) is replaced by 3 connections (the 3 material zones). The 2 output connections (skin_zone_reserve_factor selection variable and mass) are replaced by 4 connections (the 3 reserve factors and mass). Consequently, the optimization sub-workflow can be executed, resulting in an optimum material allocation for each material zone. In the next iteration, a different number of ribs will result in a different amount of material zones, requiring a change in the formulation of the sub-workflow, which is again handled by the next-design-step tool.

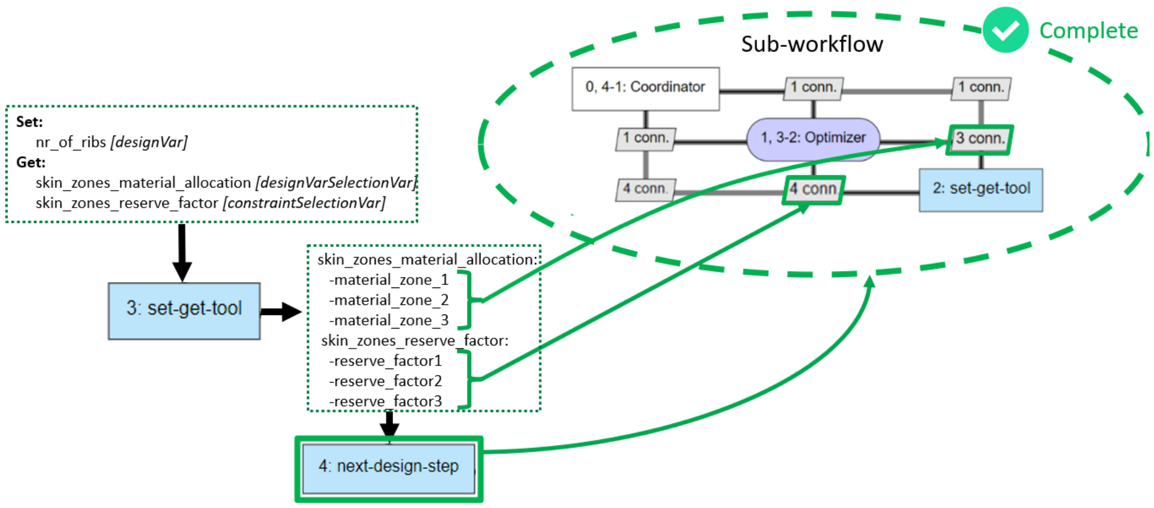


Figure 3: Dynamic reformulation of a sub-workflow in the next-design-step block, the sub-workflow is supplemented with the now known quantity of the selection variables

3. Implementation

This section describes the implementation of the methodology proposed in Chapter 2, based on the various tools and technologies provided by the DEFAINE [1] consortium. A high-level overview of the process is presented in Figure 4. The process is collaborative and distributed over three domains, the first domain is a web-based collaborative environment provided by the tool KE-chain[1], discussed in section 3.1. The second domain is a GKN Fokker computer to which the formulated workflows (discussed in section 3.2) are downloaded and converted to an executable workflow (section 3.4). During the execution, sub-workflows are dynamically formulated and executed and calls are made to a disciplinary tool for moveable modelling and structural analysis that lives on a GKN Fokker server (section 3.3). Finally, the results can be uploaded back to KE-chain for access and visualization. The various technologies used to achieve this are explained in the sections below.

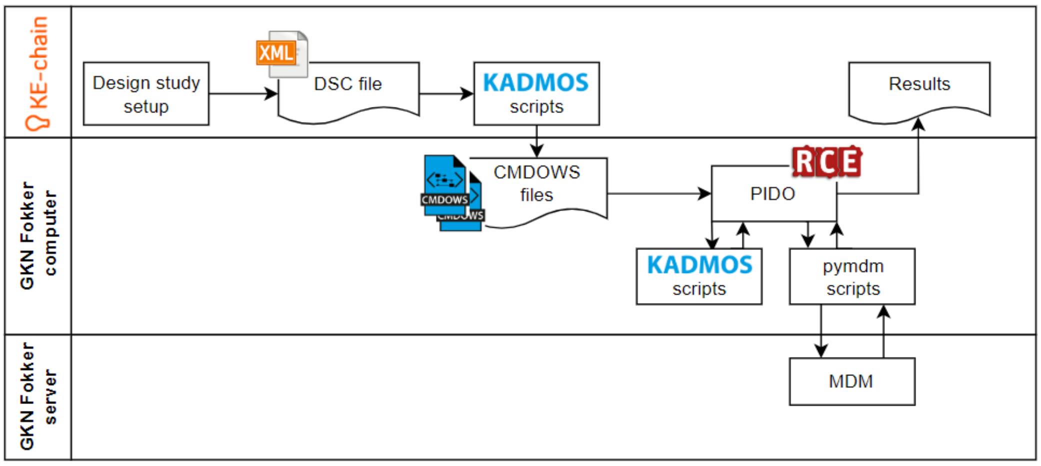


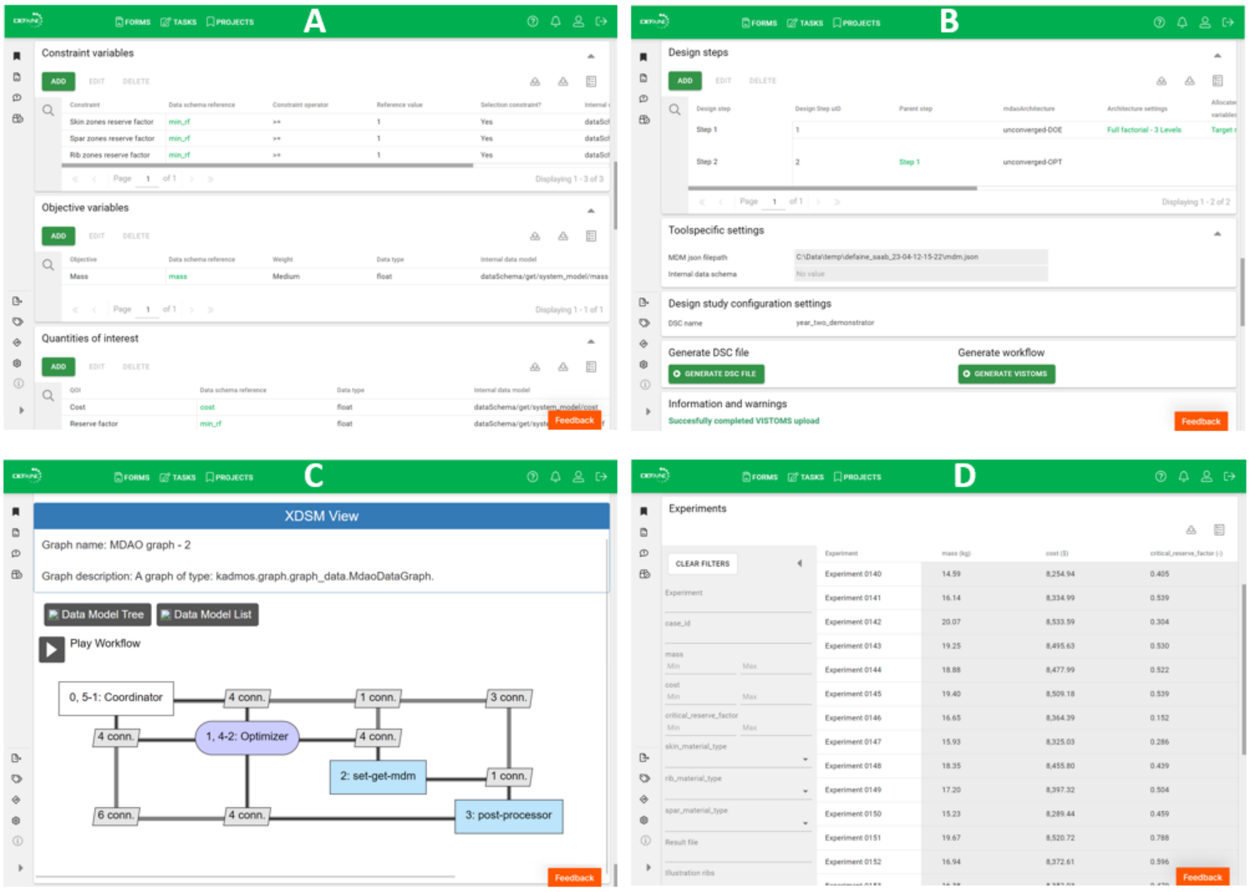
Figure 4: Overview of collaborative architecture optimization design study setup and execution process

[1] KE-chain engineering platform, KE-works, http://www.ke-chain.com/ [Accessed 27/03/2023]

3.1 Design study management using KE-chain

In the DEFAINE project, the tool KE-chain provides a web-based collaborative environment to manage the setup and execution of simulation processes. As can be seen in Figure 4, it is used to set up the design study. In addition, scripts for generating workflows (section 3.2) are hosted and can be executed on the KE-chain platform. The generated workflows can then be downloaded to a local computer on which the workflow is executed. During the execution, the results can be uploaded back to KE-chain for sharing, inspection and visualization. This is achieved by configuring the 'post-processing' block introduced in section 2.2 to write the results of a workflow iteration to KE-chain using the KE-chain python API called Pykechain[2]. Uploading results can happen both in the main and sub-workflow level.

An impression of the KE-chain web interface is shown in Figure 5. A design study can be configured by specifing the variables of interest and assigning roles (as listed in section 2.1) (5a). The hierachical relations between variables can be configured by assigning the variables to a number of design steps. By clicking a button, the DSC file and corresponding CMDOWS files (see section 3.2) can be generated and downloaded (5b). An (X)DSM representation of the formulated workflows can be visualized and inspected (5c) and during/after the execution of the design study, results can be inspected (5d).



Constraint variables

Constraint	Data schema reference	Constraint operator	Reference value	Selection constraint?	Internal d
Skin zones reserve factor	min_rf	>=	1	Yes	dataSch
Spar zones reserve factor	min_rf	>=	1	Yes	dataSch
Rib zones reserve factor	min_rf	>=	1	Yes	dataSch

Page 1 of 1 Displaying 1 - 3 of 3

Objective variables

Objective	Data schema reference	Weight	Data type	Internal data model
Mass	mass	Medium	float	dataSchema/get/system_model/mass

Page 1 of 1 Displaying 1 - 1 of 1

Quantities of interest

QOI	Data schema reference	Data type	Internal data model
Cost	cost	float	dataSchema/get/system_model/cost
Reserve factor	min_rf	float	dataSchema/get/syste

Design steps

Design step	Design Step uID	Parent step	mdaoArchitecture	Architecture settings	Allocated variables
Step 1	1		unconverged-DOE	Full factorial - 3 Levels	Target
Step 2	2	Step 1	unconverged-OPT		

Page 1 of 1 Displaying 1 - 2 of 2

Toolspecific settings

MDM json filepath C:\Data\temp\defaine_saab_23-04-12-15-22\mdm.json
Internal data schema No value

Design study configuration settings

DSC name year_two_demonstrator

Generate DSC file GENERATE DSC FILE

Generate workflow GENERATE VISTOMS

Information and warnings

Succesfully completed VISTOMS upload

XDSM View

Graph name: MDAO graph - 2

Graph description: A graph of type: kadmos.graph.graph_data.MdaoDataGraph.

Experiments

CLEAR FILTERS

Experiment
case_id
mass Min Max
cost Min Max
critical_reserve_factor Min Max
skin_material_type
rib_material_type
spar_material_type
Result file
Illustration ribs

Experiment	mass (kg)	cost ($)	critical_reserve_factor (-)
Experiment 0140	14.59	8,254.94	0.405
Experiment 0141	16.14	8,334.99	0.539
Experiment 0142	20.07	8,533.59	0.304
Experiment 0143	19.25	8,495.63	0.530
Experiment 0144	18.88	8,477.99	0.522
Experiment 0145	19.40	8,509.18	0.539
Experiment 0146	16.65	8,364.39	0.152
Experiment 0147	15.93	8,325.03	0.286
Experiment 0148	18.35	8,455.80	0.439
Experiment 0149	17.20	8,397.32	0.504
Experiment 0150	15.23	8,289.44	0.459
Experiment 0151	19.67	8,520.72	0.788
Experiment 0152	16.94	8,372.61	0.596

Figure 5: An impression of the interaction with KE-chain, the design study configuration and automatic formulation of executable workflows on-platform using the KE-chain web-based collaborative environment. (a) Adding variables and specifying roles (b) Configuring design steps, generating DSC file and downloading CMDOWS workflow files (c) Visual inspection of generated CMDOWS files. (d) Inspection of uploaded results.

[2] Pykechain Software Development Kit, KE-chain, https://pykechain.readthedocs.io/en/main/ [Accessed: 14/06/2023]

3.2 Workflow (re-)formulation using KADMOS and CMDOWS

The workflow formulation software KADMOS (Knowledge- and graph-based Agile Design for MDO Systems) [4] is used for (re-)formulating the workflows. KADMOS is a Python package developed at the Delft University of Technology that can automate the generation of MDAO formulations through a graph-based methodological approach. The XML based CMDOWS (Common MDO Workflow Schema) [5] data standard is used to store and load MDAO workflow formulations. The software is used at two locations in the process shown in figure 4, both are explained below:

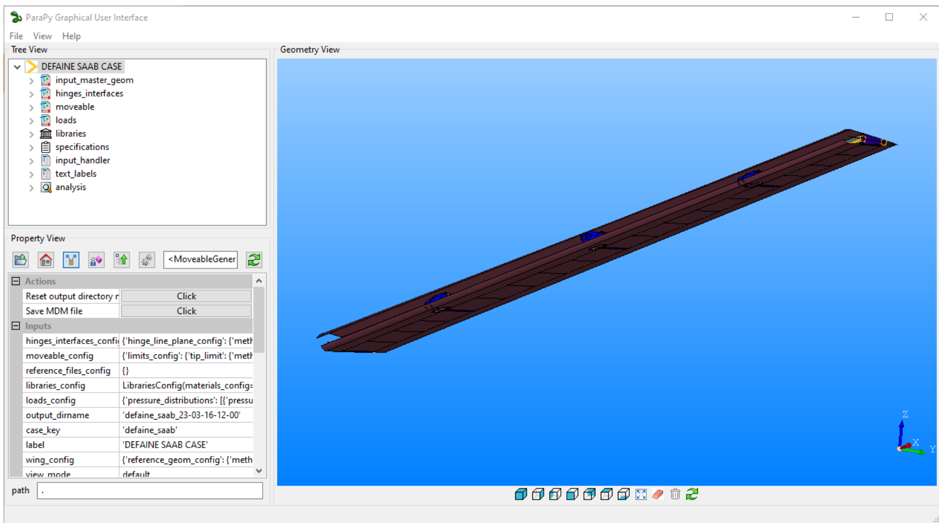
- KADMOS scripts are used to convert the DSC file into a fully formulated main workflow and, depending on the amount of specified design steps, several incomplete sub-workflow formulations, all saved as CMDOWS files.

- During execution, the CMDOWS file of the incomplete sub-workflows are read-in and finalized using KADMOS scripts within the “next-design-step" block. This is illustrated in figure 4 as a back and forth between the Process Integration & Design Optimization (PIDO) tool and KADMOS scripts, as this operation will take place each main workflow iteration.

The VISTOMS (VISualization TOol for MDO Systems) [6] package is used by KE-chain to generate (X)DSM representations of the formulated workflows stored as CMDOWS files, as shown in Figure 5d.

3.3 Interface with the disciplinary tool for moveable modelling and structural analysis

The workflows generated using the method described in section 2.2 are configured to interface with MDM [3]. MDM is a GKN Fokker tool, capable of modelling wing like aircraft components such as wings, moveables and flaps. It also provides data for and links to analysis tools that determines the performance of such aircraft components. These analyses are for example: finite element analyses, stress analysis and manufacturing cost analyses. In Figure 6, an MDM aircraft moveable model instance is shown. MDM is a Knowledge Based Engineering (KBE) application developed using ParaPy[3], made available on a dedicated Flask-server based setup and working as a web-service. An instance of the moveable is first initialized by MDM and kept alive on the server, such that ‘set’ and ‘get’ operations can be performed through http calls while the workflow is running. As some changes will only affect part of the model, the KBE system, thanks to its caching and dependency tracking [7] ensures only dependent parameters are re-evaluated, in a very computationally efficient manner. A dedicated client package ‘pymdm’ was created to interact with the server. This package provides convenient python functions that perform http calls to the server to set and retrieve slots.



DEFAINE SAAB CASE
input_master_geom
hinges_interfaces
moveable
loads
libraries
specifications
input_handler
text_labels
analysis

Reset output directory r	Click
Save MDM file	Click
hinges_interfaces_confi	{'hinge_line_plane_config': {'meth
moveable_config	{'limits_config': {'tip_limit': {'meth
reference_files_config	{}
libraries_config	LibrariesConfig(materials_config=
loads_config	{'pressure_distributions': [{'pressu
output_dirname	'defaine_saab_23-03-16-12-00'
case_key	'defaine_saab'
label	'DEFAINE SAAB CASE'
wing_config	{'reference_geom_config': {'meth
view_mode	default

Figure 6: MDM moveable model instance

[3] “Parapy Software Development Kit”, Parapy B.V., https://parapy.nl/ [Accessed 16/06/2023]

Looking at the high-level process in Figure 4, this interaction is indicated as a back and forth between the local-computer based pymdm scripts block and the server-based MDM block. It must be noted that this setup is general and can in principle be used for any KBE tool configured as a web-service.

3.4 Workflow materialization and execution using RCE

Process integration and Design Optimization (PIDO) tools facilitate the execution of MDAO workflows by enabling the integration of various analysis tools and providing built-in optimization algorithms and DOE methods. In the AGILE[4] project a plugin for DLR's PIDO tool RCE[5] has been developed, capable of converting a CMDOWS file into an executable RCE workflow. Using this plugin, the CMDOWS file representing the main workflow can be converted into an executable RCE workflow. For this to work, the previously mentioned 'initialize-tool', 'set-get-tool', 'next-design-step' and 'post-processor' blocks must be integrated into RCE. RCE will recognize the tool names in the workflow formulation and call the corresponding scripts. In Figure 7, a materialized RCE main workflow/sub-workflow combination is shown. No automatic execution is possible for the CMDOWS to RCE plugin, a manual step is always required. This is not a problem for materializing the main workflow, but poses an issue for the automatic materialization and execution of the sub-workflows. To handle this, a separate sub-workflow materialization script has been developed. This script relies heavily on the predictable format of the sub-workflow by adapting an existing template RCE optimization workflow and inserting the new design variables, constraints, objectives and quantities of interest.

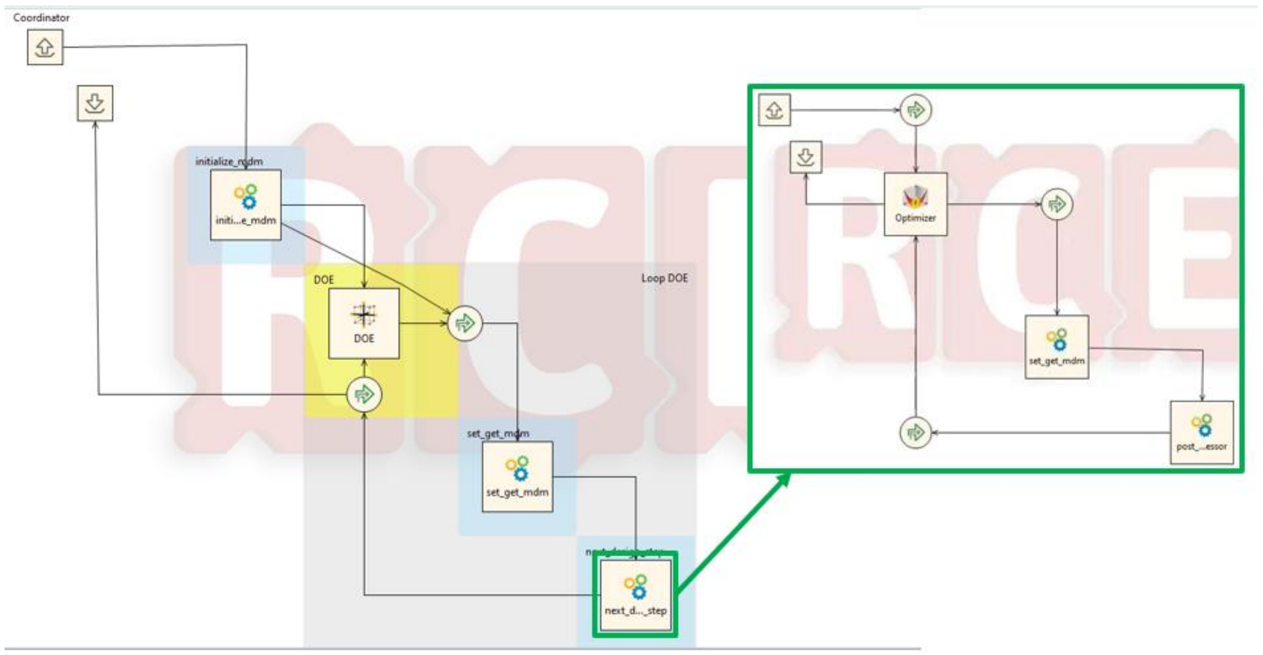


Figure 7: Executable workflows materialized in RCE from CMDOWS files, a sub-workflow is automatically generated and executed in the next-design-step block.

[4] "AGILE Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts", H2020-EU.3.4., https://www.agile-project.eu/. [Accessed 16/06/23].
[5] "Remote Component Environment", DLR, https://rcenvironment.de/ [Accessed 17/01/2023]

4. GKN Fokker Aerostructures UAV Aileron Use-case

In the DEFAINE project, GKN Fokker acts as a Tier-1 supplier for the aileron of an Unmanned Aerial Vehicle (UAV), developed by Saab, who act as Original Equipment Manufacturer (OEM). To support the OEM deciding what technologies to use and to see their effect on performance indicators such as weight and cost, the objective is to perform design space exploration and rapidly respond to OEM requests. This will be done by providing the OEM either with a dataset of sized ailerons or (a set of) response surface models. The main objectives for the aileron design are weight and cost minimization. The design variables are:

- **Number of actuators and their position**. May be specified by the OEM. The actuator position variables depend on the number of actuators specified.
- **Number of hinges and their position**. The number and position of hinges needed depends partly on the number an position of actuators.
- **Number and position of ribs**. Typically, a rib will be placed behind each hinge to transfer the corresponding loads. This results in 'bays' in which ribs can be positioned, either using a target pitch per bay or a number of ribs variable per bay. Again, these variables depend on the hinge allocation.
- **Material type** (i.e., library) selection, for example Al2024, thermoplastic composite, thermoset composite, thermoset composite with honeycomb, etc.
- **Material allocation**: the composite stack selection or metal thickness selection for a zone of the aileron parts: ribs, spars, skin panels. The number of zones to which a material can be assigned depends on the number and position of ribs and spars. More details on these 'material zones' in the MDM application, can be found in [3].

From the above, it is clear that many variables depend on the existence and value of other variables. Also, most of these variables are discrete, either integer or categorical. The main constraints are the structural analysis minimum reserve factors (RF), which should be larger than 1.0. To enable model optimization, the minimum thicknesses that satisfy the required RF are needed for each individual material zone, making the number and type of constraints also dependent on variables that change the architecture.

In the second study, the constraints of using the current implementation are removed and a problem specific 3-level nested optimization workflow was constructed by hand. This study is done to illustrate the potential of a nested optimization approach, and to show opportunities for further development beyond the current implementation.

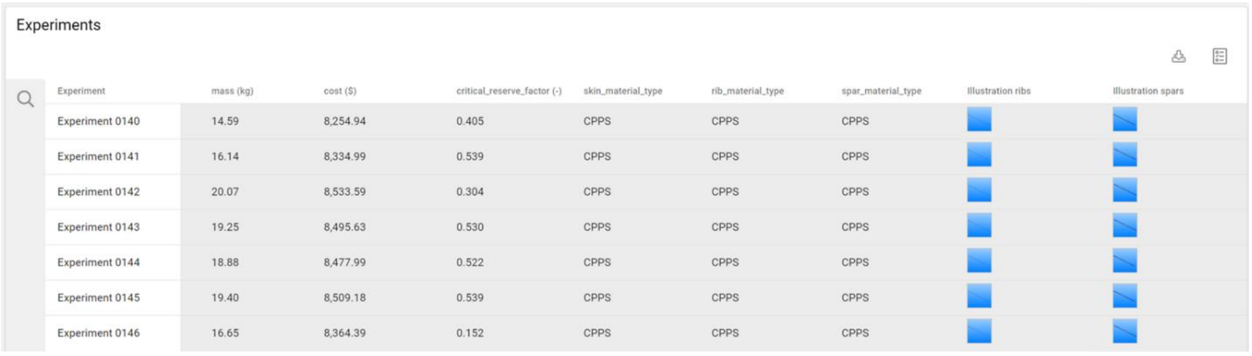
4.1 Two-level DOE Design Study

To test the system architecture optimization implementation shown in Figure 4, a simplified design study was set up, based on a two-level nested approach. As shown in Table 1, the main design step (Design step 1) consists of a Design of Experiments (DOE), with rib pitch as sole design variable. Changes in rib pitch influence the number of skin, rib and spar material zones. Because the skin, rib and spar material are configured as design variables selection variables, they are optimized in the sub-workflow (Design step 2). The sub-workflow is an optimization workflow, to minimize mass. Any moveable architecture addressed in the sub-workflow has a different rib pitch value, whose value is decided in the main workflow and remains constant in the sub-workflow. The DOE will result in optimized moveable solutions for each rib pitch, i.e., for each rib pitch, the moveable mass is minimized by changing the allocated skin, rib and spar materials.

Table 1: Design study configuration example using a main design step 1 and sub-step 2.

	Design step 1	**Design step 2**
MDAO architecture:	Design of Experiments (custom design table)	Optimization (Genetic algorithm)
Problem definition:	Quantity of interest:	Objective:
	- Cost	- Mass (minimize)
	Design variable:	Selection Variable Design variables:
	- Number of ribs	- Skin zone material allocation
		- Rib zone material allocation
		- Spar zone material allocation
		Selection Variable Constraints:
		- Skin zone reserve factor
		- Rib zone reserve factor
		- Spar zone reserve factor

Following the process described in Figure 4, a DSC file was loaded on the KE-chain platform and converted into one CMDOWS file for the main workflow and another CMDOWS file for the sub-workflow. During execution of the DOE, including the optimization of each experiment, by configuring the ‘post-processing’ block, the results are uploaded live into KE-chain, as shown in Figure 8. Unfeasible results are uploaded as well, identifiable by their RF < 1. It is useful to store unfeasible results as well, i.e. for making a surrogate model of the design space.


Experiments

Experiment	mass (kg)	cost ($)	critical_reserve_factor (-)	skin_material_type	rib_material_type	spar_material_type	Illustration ribs	Illustration spars
Experiment 0140	14.59	8,254.94	0.405	CPPS	CPPS	CPPS		
Experiment 0141	16.14	8,334.99	0.539	CPPS	CPPS	CPPS		
Experiment 0142	20.07	8,533.59	0.304	CPPS	CPPS	CPPS		
Experiment 0143	19.25	8,495.63	0.530	CPPS	CPPS	CPPS		
Experiment 0144	18.88	8,477.99	0.522	CPPS	CPPS	CPPS		
Experiment 0145	19.40	8,509.18	0.539	CPPS	CPPS	CPPS		
Experiment 0146	16.65	8,364.39	0.152	CPPS	CPPS	CPPS		

Figure 8: KE-chain web interface screenshot showing uploaded results of a design study

This initial implementation proved effective in handling the hierarchal and mixed-integer nature of the design space, and, thanks to the various implemented technologies (such as KE-chain, KADMOS, CMDOWS etc.) is expected to reduce the time needed for setting up and executing design studies in a collaborative environment. However, RCE has a limited selection of build-in optimization algorithms capable of handling integer variables, and none that can handle categorical variables. A workaround was found by using a mapping file to convert string values to integers and using a genetic optimization algorithm to deal with the integer design variables. However it was not possible to conduct a full optimization of both levels. Therefore, a manually configured DOE was set up for the main workflow, and the genetic algorithm was used to evaluate a limited number of design points in the sub-workflow, just to prove the concept of formulating nested architectures supported by dynamic workflow capability.

In the next section, an alternative implementation is described, where a full multi objective architecture optimization of the same aileron was completed. This implementation is however based on manual formulation of the problem and not yet making use of the aforementioned technologies to enable collaborative design.

4.2 Three-level Multi-objective Optimization

To illustrate the potential of the nested optimization approach proposed in this paper for system architecture optimization, another workflow implementation was tested on the same wing moveable system. An overview of the process is shown in Figure 9. The key difference with the implementation discussed in Section 4.1 (Figure 4) is the use of the open-source Python optimization framework pymoo [10], in place of RCE. Besides that, the formulation was manually implemented and not based on the DSC approach or any of the technologies for collaborative design discussed in 3.1 and 3.2. Here, pymoo was employed to create and run the outer optimization loops, as well as to formulate the sub-problems stemming from upstream changes in the architectural variables. The aileron KBE model was accessed through the pymdm package in the same way as described in Section 3.3.

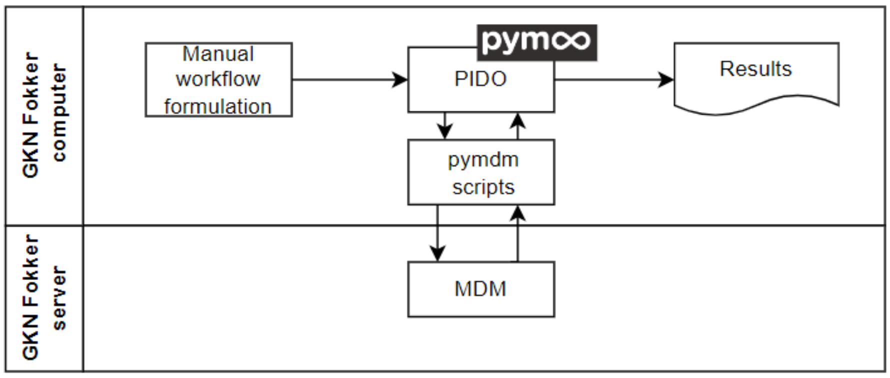


Figure 9: overview of the manually set up multi-objective optimization study.

This problem is formulated as a multi-objective optimization, aiming at minimizing both mass and cost of the moveable. The involved design variables are grouped into three levels according to their hierarchical relationships (Table 2). The variables at the outermost level are the number of hinges and the material libraries for the moveable's ribs, spars and skins. Based on the number of hinges set at the outermost level, the middle level establishes their spanwise positions. The middle level is also responsible for setting the number of secondary ribs in each bay delimited by the main ribs (i.e., the ribs at the root and tip of the moveable and the ribs behind actuators and hinges). Finally, the innermost level allocates the material at each resulting material zone, given each component's library set in the outermost level.

Table 2: Summary of the three-level multi-objective optimization problem for aileron architecture design studies.

	Step 1 (architecture definition)	**Step 2 (sub-step)**	**Step 3 (material allocation)**
MDAO architecture:	Optimization (NSGA-II)	Optimization (NSGA-II)	Constraint satisfaction sizing (custom direct search algorithm)
Problem definition:	Objectives: - Mass (minimize) - Cost (minimize) Architecture design variables: - Number of hinges - Skin material library - Rib material library - Spar material library Constraints: - Overall reserve factor	Objectives: - Mass (minimize) - Cost (minimize) Design variables: - Hinge spanwise locations - Number of ribs at each bay - Spar zone material allocation Constraints: - Overall reserve factor	Quantities of interest: - Mass - Cost Design variables: - Skin material zone allocation - Rib material zone allocation - Spar material zone allocation Constraints: - Overall reserve factor

The NSGA-II algorithm [11] was employed at the two top optimization levels to adapt the procedure to multiple objectives. The innermost level in the optimization is driven by a custom direct search algorithm that defines material zone allocations that satisfy the minimum reserve factor constraint. Figure 9 summarizes the complete nested optimization procedure.

At each level, the optimizer only knows the design variables related to its subproblem. As a result, the upper-level optimizers might encounter identical combinations of design variables that yield different objective and constraint values. However, because the principle behind these top-level optimizers is non-dominated sorting, this feature can be exploited to set the budget for searching the most promising areas in the design space.

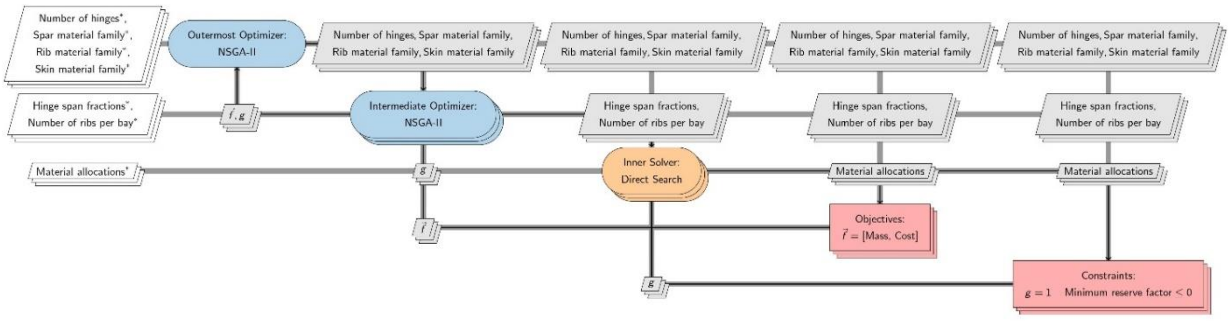

Number of hinges*, Spar material family*, Rib material family*, Skin material family*
Outermost Optimizer: NSGA-II
Number of hinges, Spar material family, Rib material family, Skin material family
Number of hinges, Spar material family, Rib material family, Skin material family
Number of hinges, Spar material family, Rib material family, Skin material family
Number of hinges, Spar material family, Rib material family, Skin material family
Hinge span fractions*, Number of ribs per bay*
$\vec{f}, g$
Intermediate Optimizer: NSGA-II
Hinge span fractions, Number of ribs per bay
Hinge span fractions, Number of ribs per bay
Hinge span fractions, Number of ribs per bay
Material allocations*
g
Inner Solver: Direct Search
Material allocations
Material allocations
$\vec{f}$
Objectives: $\vec{f}$ = [Mass, Cost]
g
Constraints: $g = 1$ Minimum reserve factor < 0

Figure 9: XDSM representation of the three-level multi-objective optimization procedure.

The multi-objective optimization was run by setting a population size of 35 at the outermost level and six at the middle level. The maximum number of iterations for the innermost level's search algorithm was set to 50. The optimization results after four generations of the outermost loop are shown in Figure 10.

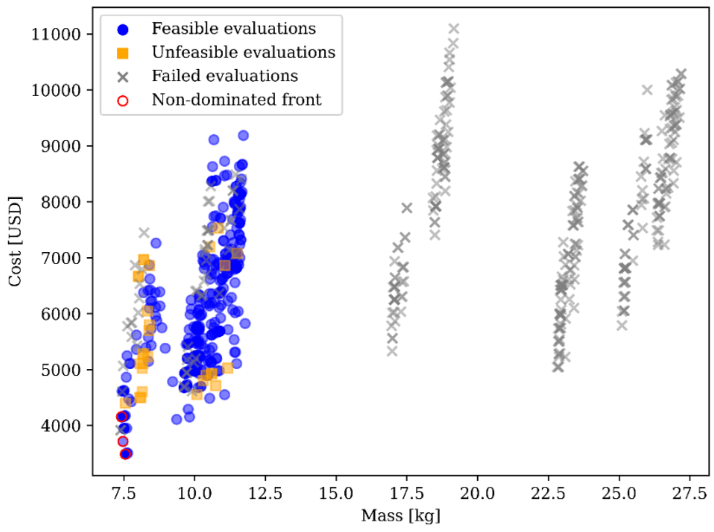


Figure 10: Optimization results after four generations of the outermost loop. Feasible evaluations have a minimum reserve factor greater or equal to 1; unfeasible evaluations have a minimum reserve factor lower than one; failed evaluations were unsuccessful in calculating the reserve factor.

During the optimization, the evaluation of the minimum reserve factor constraint failed for some design points. The cause of these failures is still under investigation. Unforeseen failures in computational tools can be seen as "hidden constraints" that limit the access of portions of the design space to the optimizer [12]. The optimization routine did find a feasible region with three non-dominated points.

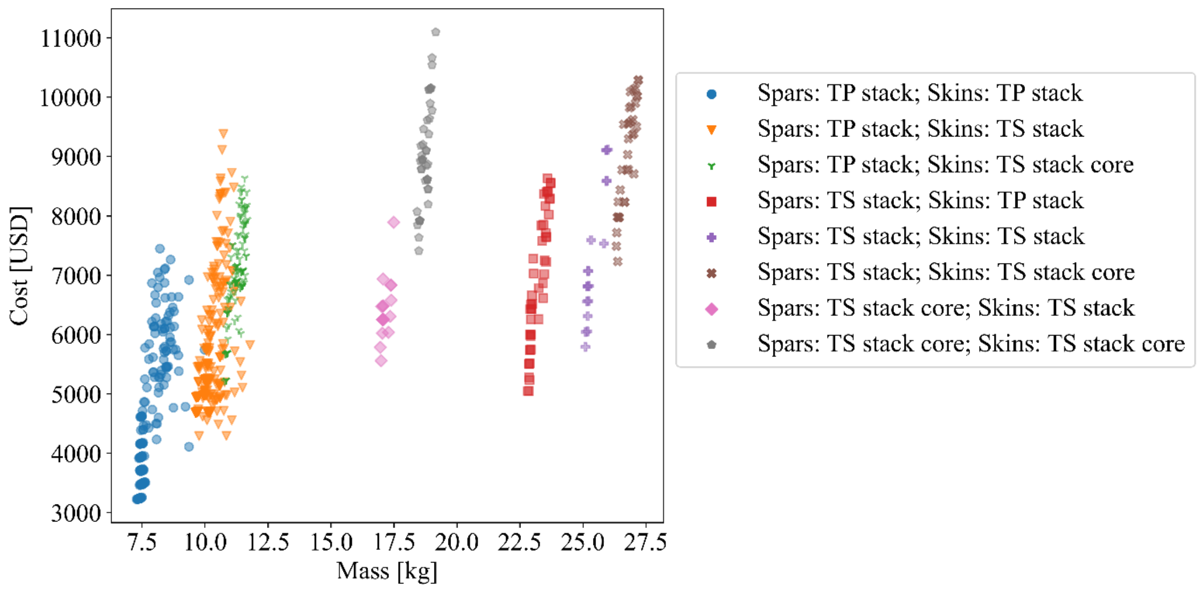


Figure 11: Points in the objective space colour marked according to spar and skin material. TP, TS and core stand respectively for thermoplastic, thermoset and inner honeycomb structure.

An interesting feature of Figure 10 is the groupings of different design points in clusters of architectures with similar mass values, stretched along the cost axis. As shown by colour coding used in Figure 11, these clusters are explained by the combination of material library selections for the moveable's skins and spars. The material selection is performed at the outermost (or architectural) loop in the optimization run, highlighting the importance of including architectural decisions in the design process.

5. Conclusions and Outlook

In this paper, a methodology was presented to address system architecture optimization problems, where hierarchical relations exist between design variables, i.e. where the quantity and type of some variables depend on the value assumed by “higher-level” variables, in a way not known a priori. To tackle this challenge, a nested workflow approach was proposed, where an outer loop control the high-level variables, based on which a nested loop is dynamically formulated and executed, which in turn can lead to other dynamically formulated nested loop and so on.

The methodology was implemented in the scope of the project DEFAINE, making use of technologies provided by the consortium to enable collaborative design exploration studies of aeronautical systems. These technologies include a web based tool to manage multi-actor, multi-site collaborative projects (KE-Chain), a tool for automatic formulations of MDAO workflows (KADMOS), a data schema to store and distribute MDAO problem formulations (CMDOWS), a data schema to configure nested workflows based on dynamic formulations (DSC), a KBE tool for wing moveables design and multidisciplinary analysis (MDM) and an open source PIDO tool (RCE). A first implementation of the proposed methodology was tested on a GKN-Fokker aileron architecture optimization use-case. The approach proved effective in dealing with the hierarchical mixed-integer design space and enabled a flexible set up of a nested workflow, consisting of an outer DOE loop and an inner, dynamically formulated, loop for structural optimization. Collaborative trade off studies of various aileron architectures could be set in a time efficient manner due to the high level of achieved process automation.

Limitations in the optimization capabilities of the selected PIDO tool, however, prevented the execution of full architecture optimization studies. Therefore, a second implementation of the proposed nested approach was manually assembled and executed on a local installation at GKN-Fokker, using pymoo in place of RCE and none of the collaborative technologies exploited in the first implementation. A 3-level nested multi-objective optimization was assembled and successfully executed, demonstrating the capability of the proposed nested approach to address complex architecture optimization problems. The study offered GKN-Fokker useful insights on the impact of certain architectural choices, such as material allocation, number of ribs and hinges on the mass and cost of many aileron design variants.

As the intent of DEFAINE is to perform such architectural design exploration and optimization studies in a collaborative environment, the following developments are ongoing:

- The DSC format was necessary because of the CMDOWS limits to support the specification of sub-workflows and hierarchical variables within MDAO workflows. At the time of writing, CMDOWS is being extended to include sub-workflows. At the same time, KADMOS is being updated to interpret the extended CMDOWS schema and formulate dynamic workflow accordingly.
- RCE workflows can be generated almost automatically, thanks to a plug-in developed during the AGILE project, to parse CMDOWS files. At this moment this plug-in is not actively supported and some manual work is still require to materialize the executable workflow. Considering also the inherent limitations in the suite of optimization algorithms provided by RCE, a new CMDOWS plug-in being developed targeting commercial, state-of-the art PIDO tools.
- The material libraries used in the two implementations discussed in the paper are limited to composites. The inclusion of other materials such as metal, would require the capability to dynamically configure workflows, not only with different design variables, but also with different constraints and analysis tools. To this purpose, a connection to the Requirements Verification Framework (RVF) described in [13] is in progress. The RVF is capable of deriving MDAO workflows from stakeholder requirements, ensuring the required disciplinary tools are embedded in the workflow, according to system architecture under consideration.

Acknowledgements

The research presented in this paper has partially been performed in the framework of the DEFAINE (Design Exploration Framework based on AI for front-loaded Engineering) project and has received funding from ITEA 3 programme (Call 6 project 19009).

References

[1] M. Baan, et. al., “DEFAINE – Design Exploration Framework based on AI for front-loaded Engineering: Achievements and Open Challenges”, submitted to Joint 10th EUCASS-9th CEAS Conference, Lausanne, 2023
[2] J.H. Bussemaker, T. De Smedt, G. La Rocca, P.D. Ciampa, B. Nagel. “System Architecture Optimization: An Open Source Multidisciplinary Aircraft Jet Engine Architecting Problem”, AIAA Aviation Forum, 2021
[3] T. van den Berg, B. van Manen, A.H. van der Laan, I. Ciobotia, D. Bansal, J.S. Sonneveld, “A multidisciplinary modeling system for aircraft structural components”, submitted to Joint 10th EUCASS-9th CEAS Conference, Lausanne, 2023
[4] I. van Gent, G, La Rocca, “Formulation and integration of MDAO systems for collaborative design: A graph-based methodological approach”, Aerospace Science and Technology, 2019
[5] I. van Gent, G. La Rocca, M.F.M. Hoogreef, “CMDOWS: a proposed new standard to store and exchange MDO systems”, CEAS Aeronautical Journal, Vol. 9, No. 4, May 2018, pp. 607–627.
[6] B. Aigner, I. Van Gent, G. La Rocca, E. Stumpf and L. L. M. Veldhuis, “Graph-based algorithms and data-driven documents for formulation and visualization of large MDO system,” CEAS Aeronautical Journal, vol. 9, pp. 695-709, 2018.
[7] G. La Rocca. “Knowledge based engineering: Between AI and CAD. review of a language based technology to support engineering design”, Advanced engineering informatics, 26(2):159–179, 2012
[8] C. Frank, R. Marlier, O. J. Pinon-Fischer, and D. N. Mavris, “An Evolutionary Multi-Architecture Multi-Objective Optimization Algorithm for Design Space Exploration,” 57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, American Institute of Aeronautics and Astronautics, Jan. 2016, pp. 1–19.
[9] J.J. Michalek and P.Y. Papalambros, “BB-ATC: Analytical Target Cascading Using Branch and Bound for Mixed-Integer Nonlinear Programming,” Volume 1: 32nd Design Automation Conference, Parts A and B, ASMEDC, Jan. 2006, pp. 1–7. doi:10.1115/detc2006-99040.
[10] J. Blank and K. Deb, "Pymoo: Multi-Objective Optimization in Python," in IEEE Access, vol. 8, pp. 89497-89509, 2020
[11] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," in IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, April 2002

[12] J.H. Bussemaker, N. Bartoli, T. Lefebvre, P.D. Ciampa and B. Nagel. "Effectiveness of Surrogate-Based Optimization Algorithms for System Architecture Optimization," AIAA 2021-3095. AIAA AVIATION 2021 FORUM. August 2021.
[13] A.M.R.M. Bruggeman, B. van Manen, T. van der Laan, T. van den Berg and G. La Rocca, “An MBSE-Based Requirement Verification Framework to support the MDAO Process”, AIAA Aviation Forum, 2022.