D4.1.3. Release of Workflow (re-) formulation tool(s)

Author, company:
- Jente Sonneveld, TU Delft
- Anne-Liza Bruggeman, TU Delft
- Gianfranco La Rocca, TU Delft

Version:
1.0

Date:
February 7, 2024

Status:
Final / Released

Confidentiality:
Public

Change log

Revision	Date	Prepared by	Checked by	Description
0.1	22/11/2023	Jente Sonneveld	Anne-Liza Bruggeman	First setup document
0.2	25/01/2024	Anne-Liza Bruggeman	Gianfranco La Rocca	Added part on sub-workflows and switches in KADMOS
0.3	07/02/2024	Gianfranco La Rocca	Bastiaan Beijer	Review and editing

Table of Contents

Acronyms

Acronym	Definition
CMDOWS	Common MDO Workflow Schema
DEFAINE	Design Exploration Framework based on AI for froNt-loaded Engineering
DOE	Design Of Experiments
KADMOS	Knowledge- and graph-based Agile Design for Multidisciplinary Optimization System
MDAO	Multidisciplinary Design Analysis and Optimization
MDO	Multidisciplinary Design Optimization
PIDO	Process Integration and Design Optimization
WP	Work Package
XDSM	eXtended Design Structure Matrix
XML	eXtensible Markup Language

Acknowledgements

This research is partly funded by the ITEA 3 Call 6 project DEFAINE of the European Union.

References

[1] AGILE, “AGILE Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts,” H2020-EU.3.4., [Online]. Available: https://www.agile-project.eu/. [Accessed 23 12 221].

[2] AGILE4, “AGILE4.0 Towards cyber-physical collaborative aircraft development,” H2020 No 815122, [Online]. Available: http://agile4.eu. [Accessed 23 12 2021].

[3] DEFAINE Consortium, “D4.1.2. Second release of Workflow (re-)formulation tool(s),” ITEA3 Call 6, 2022.

[4] DEFAINE Consortium, “D4.3.1. Technology demonstrator for use-cases,” ITEA3 Call 6, 2022.

[5] DEFAINE Consortium, “D4.3.2. Second release of the techonlogy demonstrator for use-cases,” Itea3 Call 6, 2022.

[6] DEFAINE Consortium, “D4.3.3. Final release of the techonlogy demonstrator for use-cases,” Itea3 Call 6, 2023.

[7] DEFAINE Consortium, “D4.1.1. First release of Workflow (re-)formulation tool(s),” ITEA3 Call 6, 2022.

[8] DEFAINE Consortium, “D4.2.1. CMDOWS,” ITEA3 Call 6, 2022.

[9] A. B. Lambe and J. R. R. A. Martins, “Extensions to the design structure matrix for the description of multidisciplinary design, analysis and optimization processes,” *Structural and Multidisciplinary Optimization,* vol. 46, pp. 273-284, 2012.

[10] J. Sonneveld, T. van den Berg, G. la Rocca, S. Valencia Ibanez, B. van Manen, A. Bruggeman and B. Beijer, “Dynamic workflow generation applied to aircraft moveable architecture optimization,” in *CEAS*, Lausanne, 2023.

[11] DEFAINE Consortium, “D2.1.1. Use-case specifications,” ITEA3 Call 6, 2021.

1. Introduction

Work package (WP) 4 aims to achieve automated (re-)formulation of workflows. This deliverable describes the released workflow (re-)formulation tool(s). In an industrial environment, the setup of Multidisciplinary Design Analysis and Optimization (MDAO) workflows usually requires manual intervention, which is time and cost intensive, as well as prone to human errors. This frustrates the ideal MDAO process, where designers are typically interested in formulating first simple Multidisciplinary Design Analysis (MDA) workflows, then performing Design of Experiments (DOEs) and sensitivity studies to identify relevant design parameters, before moving to actual optimization and eventually iterate on the previous steps, i.e. to add/remove design variables and constraints, test different objectives or add/substitute some of the analysis tools. Previous efforts in the IDEALISM[1], AGILE [1] and AGILE4.0 [2] projects have produced methodologies and tools to provide some of the required agility in MDAO workflow (re-)formulation and execution. However, dynamic re-formulations of workflows during the design process are not yet possible using State-of-the-Art methods and tools.

By “dynamic re-formulation”, we intend the capability of a given workflow to reformulate itself, i.e. to adjust, among others, the involved design variables, the constraints, as well as the involved design competences (analysis tools) at execution time. This goes beyond the advisory capabilities addressed in D4.1.2 [3], where MDAO workflow reformulations were suggested (and possibly automatically implemented) before executing the actual MDAO process. As DEFAINE aims at performing extensive design space explorations, dynamic re-formulations capabilities are pursued to address challenges in architecture design space exploration by enabling the analysis of multiple architectures in a single (dynamically adapting) MDAO workflow.

During the DEFAINE project, the workflow (re-) formulation methodologies and tool(s) are developed and released in three cycles, aligning with the release of the technology demonstrators described in D4.3.1 [4], D4.3.2 [5] and D4.3.3 [6]. This will be done by further developing the open-source software KADMOS for MDAO system formulations developed in the AGILE [1] and AGILE4.0 [2] projects. The tool released in this cycle is the third and final release of this deliverable, in Table 1, all releases of this deliverable are shown.

Table 1: Releases of workflow (re-)formulation tools deliverables.

Deliverable	Description	Due
D4.1.1	State of the Art	M16 (delivered)
D4.1.2	Extend the existing advisory capabilities	M28 (delivered)
D4.1.3	Enable dynamic reformulation capabilities **(this deliverable)**	M40

[1] IDEaliSM (uni-stuttgart.de), accessed on: 25-01-2024

1.1. Intended use and purpose of this deliverable

This deliverable is of type “software”. The purpose of this deliverable is to provide an overview of all the developments achieved after the release of the previous version of this deliverable (D4.1.2) [3]. This deliverable describes developments towards dynamic workflows.

In the original DEFAINE project proposal dynamic re-formulation capabilities were proposed as a means to drastically reduce computational time, by making workflow formulations more efficient. The idea was to create capabilities for adjusting the workflow formulation at run time, hence reacting on information gathered while running the simulation. The envisioned reformulation included removal of low sensitivity design variables and inactive constraints, change of the MDO system architecture, parallelization of workflows, switch of tools according to license availability or necessary level of fidelity. In response to the needs expressed by DEFAINE industrial partners and especially to the insights developed during the research work, the focus has moved towards the combined development of advisory capabilities (see the KADMOS extensions to enable automated workflow partitioning, global/local sensitivity analysis, and the surrogate model advisory system addressed in D4.1.1 [7] and 4.1.2 [3]) and dynamic workflows reformulation capabilities, specifically addressing challenges in architecture design space exploration.
The developments described in this deliverable include the following KADMOS extensions:

- Introduction of the new ‘sub-workflow’ executable block (to enable the formulations of MADO workflows, where disciplinary blocks can be MDAO workflows themselves
- Introduction of new ‘switch’ block, to enable alternative workflows branches to be executed within the same exploration/optimization process
- Introduction of new capability to allow part of a MDAO workflow (i.e. a sub-workflow) to be dynamically formulated during the execution of the main MDAO workflow.

The 3 developments listed above are presented in Chapter 2. Chapter 3 provides some overall conclusions.

2. Dynamic workflows

A definition of dynamic workflow is provided in Chapter 1. Below, two scenarios for the use of dynamic reformulating workflows are described.

The first scenario concerns the need to activate/deactivate some elements of the MDAO workflow, depending on the design point being analysed. This is the case when different design variables, disciplinary tools or constraints are required to evaluate one concept (e.g., a certain product architecture) instead of another. If one wants to trade these two alternative concepts by means of one MDAO workflow, different parts of the workflow needs to be activated and deactivated depending on the concept being analysed. In other words, the MDAO workflows need to dynamically change by switching between concepts. For the sake of rigour, in this case, the MDAO system is not reformulated, but its execution process is dynamically adapted during the same optimization run. Indeed, the MDAO system is formulated just once, including alternative branches to be triggered according to the current design vector.

In a second scenario, dynamic reformulation can be utilized to enable the execution of a workflow where some of its elements are not known beforehand (i.e. at the time of the formulation), but only discovered during execution time. A typical instance of this challenge arises in the presence of hierarchical variables, where the type and quantity of some variables are dependent upon the value assigned to another variable. In addition, the analysis module(s) to be called upon might also depend on the value assumed by some of the variables (similarly to the first scenario). In these cases, depending on the value of certain variables, part of the workflow must be reformulated during execution.

The next sections describe the solutions developed in DEFAINE to address both scenarios. First, the general concept of sub-workflows and its implementation in KADMOS will be explained, as this concept is a key enabler of the dynamic MDAO workflows. Next, the newly implemented switch will be presented. The switch takes care of activation and deactivation of design variables, disciplines and constraints during the same exploration/optimization process, depending on the product architecture being analysed at a certain iteration. Finally, the implementation of dynamic reformulation workflows (scenario 2) is discussed, addressing the proposed concept of *placeholder variables*.

2.1. Sub-workflows

The concept of sub-workflow is a key enabler for dynamic MDAO workflows. A sub-workflow can be defined as a workflow in a workflow. A sub-workflow is treated (and visualized) in the same manner as a ‘normal’ disciplinary tool, as shown in the example in Figure 1. In this case, the workflow on the left is the main workflow. This workflow contains a discipline called Manufacturing. The Manufacturing discipline is in fact a workflow itself, i.e. a sub-workflow, in which a manufacturing optimization is performed, involving the Machining disciplinary tool (Figure 1, right).

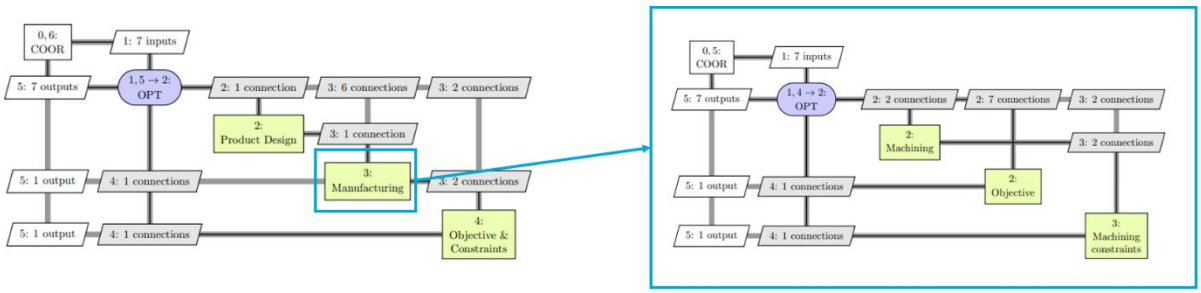


Figure 1: Example of a sub-workflow in a main workflow. In this case, the manufacturing block in the main workflow (left) is a manufacturing optimization sub-workflow (right)

KADMOS has been expanded to support the formulation of workflows containing sub-workflows. The sub-workflows have been implemented in a similar way as the design competences and mathematical functions. The main difference is that the sub-workflow nodes contain an extra attribute called 'graph'. Here a new KadmosGraph is stored that represents the sub-workflow.

As the sub-workflow is a workflow on its own, the user must now also give inputs that are specific for each sub-workflow. This input includes the tools that that must be included in the sub-workflow, the design variables, constraints, objectives and quantities of interests, and the MDAO solution strategy to implement (e.g. Optimization, DoE, MDA). All functions (e.g. getting the function order, applying the MDAO strategy, etc.) have been adapted such that they are automatically applied also to the sub-workflows.

Besides KADMOS, also CMDOWS, the data schema used to store the KADMOS generated MDAO system formulations, has been adapted accordingly. Additionally, a Python-based plug-in has been developed to parse the generated CMDOWS files and automatically translate them into Optimus MDAO workflows, ready for execution. The details about these developments can be found in D4.2.1 [8]. The reason why it was opted to target the commercial PIDO tool Optimus, by Noesis is briefly discussed in the next sub-section.

2.2. Switches and extended XDSM

One of the scenarios that requires dynamic MDAO workflows is when different product concepts need to be traded within the same MDAO workflow. An example is shown in Figure 2.

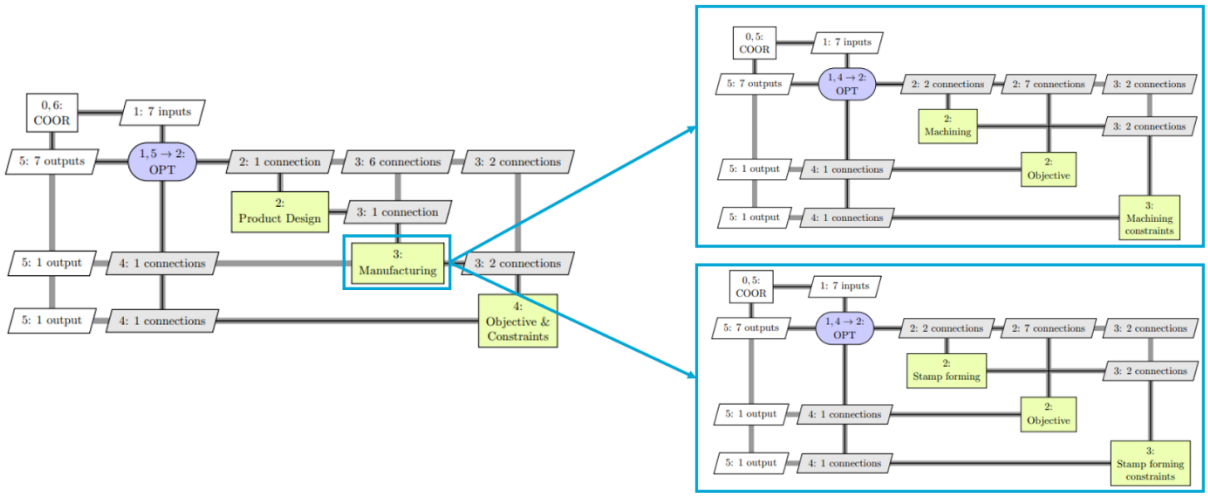
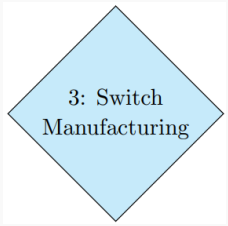
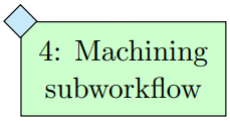


Figure 2: Example scenario in which a switch is required. In this case, the manufacturing block can be represented either by a machining sub-workflow or a stamp forming sub-workflow, depending on which production method is chosen in the main workflow. Depending on the production method, different design variables, disciplinary tools and constraints must be used.

Here, the product of interest (e.g. a wing rib) can be produced using either metal machining or composite stamp forming. The manufacturing process to be used is in this case a design variable. Depending on to the production method selected by the optimizer at a given iteration, , different design variables, analysis disciplines and constraints need to be activated or deactivated.

To enable this evaluation, a switch with branches has been introduced in KADMOS. A switch activates and deactivates different branches depending on the input it gets and the conditions for each branch. A branch consists of a design competence, mathematical equation or a sub-workflow (see Section 2.1). To add a switch to KADMOS, the user only has to specify the alternative (and mutually exclusive) branches, together with their activation conditions. A condition could be for example “production_method=machining”.

In order to provide convenient visualizations of such dynamic workflows, in this work an extension to the XDSM has been proposed. The following new icons have been created:

	The switch is represented by a blue diamond. The diamond shape has been chosen as a decision is generally represented by a diamond in flowcharts. The colour blue has been chosen as this one was not yet used by any other icon in the XDSM.
	A branch is represented like a normal discipline, with the addition of a small blue diamond in the upper left corner. A branch is always placed next to a switch and the small diamond indicates the discipline is activated by that switch.

4: Part A Manufacturing Method Subworkflows	As one switch can have many different branches, a new visualization has been created to represent the branches in a compact way (if required by the user). A stacked icon of disciplines has been chosen, again with a small blue diamond in the upper left corner. The stacked representation in the XDSM is generally used for disciplines that are executed in parallel. The branches of the switch are also parallel disciplines (they do not depend on each other). However, the branches are mutually exclusive, meaning that only one of them can be selected for executed, per iteration.

Figure 3 shows an example XDSM for an MDAO system, based on the MDF Gauss-Seidel architecture, including the newly introduced concept of switch. The terminology used in this figure is similar to the terminology used by Lambe and Martins [9], the original proposers of the XDSM. Three new concepts have been introduced in this XDSM visualization:

- d: decision variables
  Decision variables are Booleans that indicate whether a branch must be executed or not. The switch evaluates the conditions for each branch and if all conditions are met, the decision variable for that branch is set to True. Note that the branches are mutually exclusive, which means that only one decision variable can be set to True per iteration.
- $()_b$: applies to branch
  The subscript b indicates that the variable only applies to the branches of a switch. This could either be an input variable or an output variable.
- $()_s$: applies to switch
  The subscript s indicates that the variable only applies to the switch. As the output for the switch are decision variables, only input variables can have the subscript s.

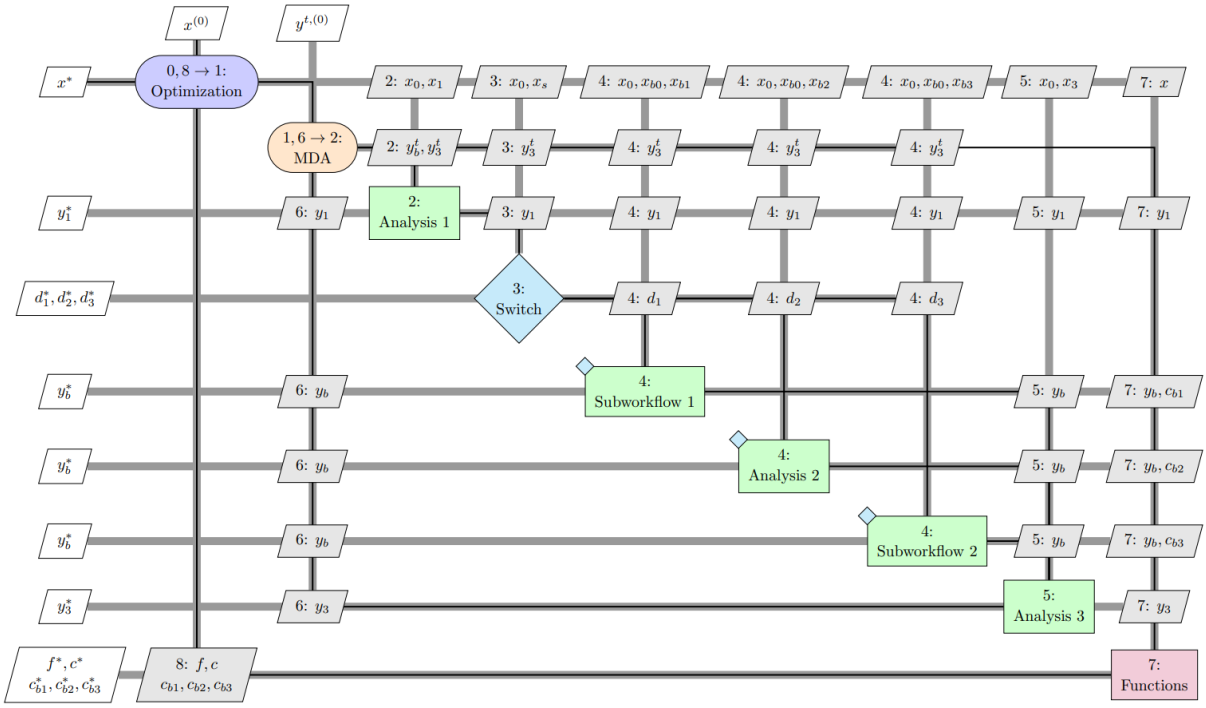


Figure 3: General visualization of the switch implemented in an MDF Gauss-Seidel MDAO architecture

The switch has been implemented into the PIDO tool Optimus. The reason to implement the switch in Optimus is because this tool natively supports the concepts of switches and sub-workflows (although not originally developed to enable dynamic workflows). The development of a custom output interface was necessary to enable the results of a sub-workflow in a switch to be fed back to the main workflow. An example of an Optimus workflow with a switch and sub-workflows is shown in Figure 4.

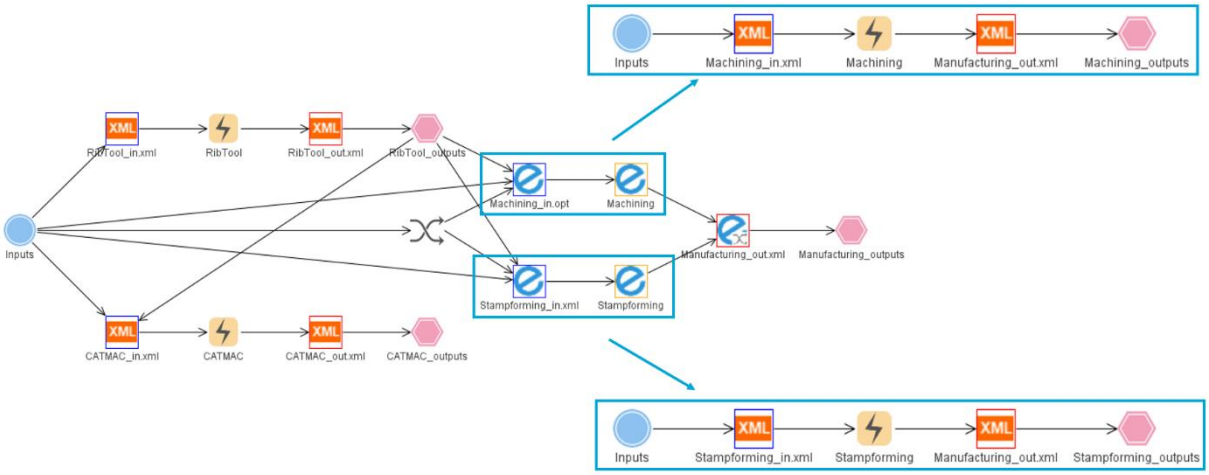


Figure 4: Example of an Optimus workflow including a switch and sub-workflows. The sub-workflows are implemented using the Opt-In-Opt function (represented by the blue "e"s in the main workflow). The switch is represented by the crossing arrows icon

2.3. Dynamic reformulating sub-workflows

In architecture design exploration and optimization, hierarchical variables are often present, meaning that the existence, quantity and type of certain variables depend on the value assumed by other variables. In some cases, the effect of a certain high level architecture variable on the quantity of its dependent variables may not be known a priori. This makes it impossible to fully formulate an MDAO problem featuring such dependent variables. This scenario is sketched in Figure 5 for the design of a moveable, where two top-level architecture variables 'Nr of ribs' and 'Material' dictate the amount and type of the dependent variables 'material zone'.

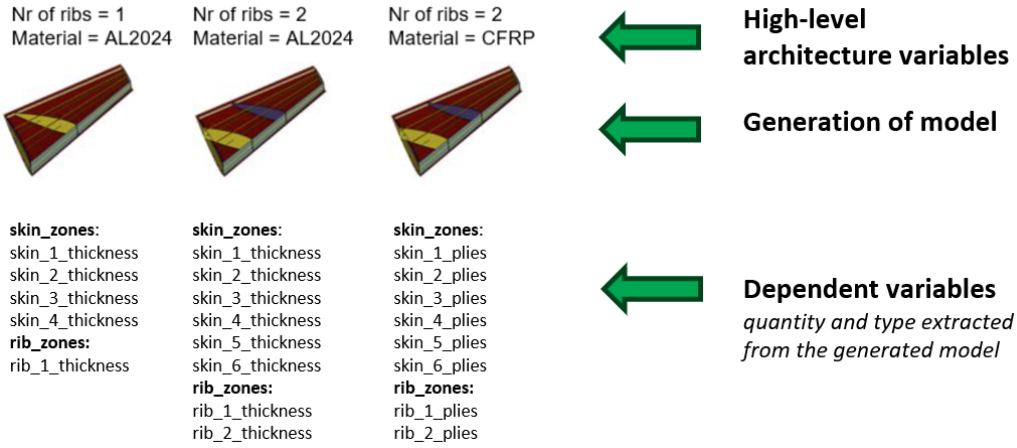


Figure 5: Example of hierarchical relationship between variables in a moveable design

In D4.1.2 [3] and [10], an approach for architecture design exploration and optimization was presented to deal with the above-mentioned problem. The approach has further matured and is now aligned with the implementation of the 'sub-workflow' type within KADMOS.

The approach splits the problem such that high-level architecture variables are placed in a 'main-workflow' and their dependent variables are handled in a nested sub-workflow. At each iteration, a KADMOS-based dynamic re-formulation script is used to complete the sub-workflow formulation. This concept is illustrated in Figure 6. In this example, a single KBE tool is used to handle all input and output requests. Once the tool is initialized using the 'initialize_tool' discipline, the interaction is handled by the 'set_get_tool' discipline. The same 'set_get_tool' is used in both the main- and sub-workflow.

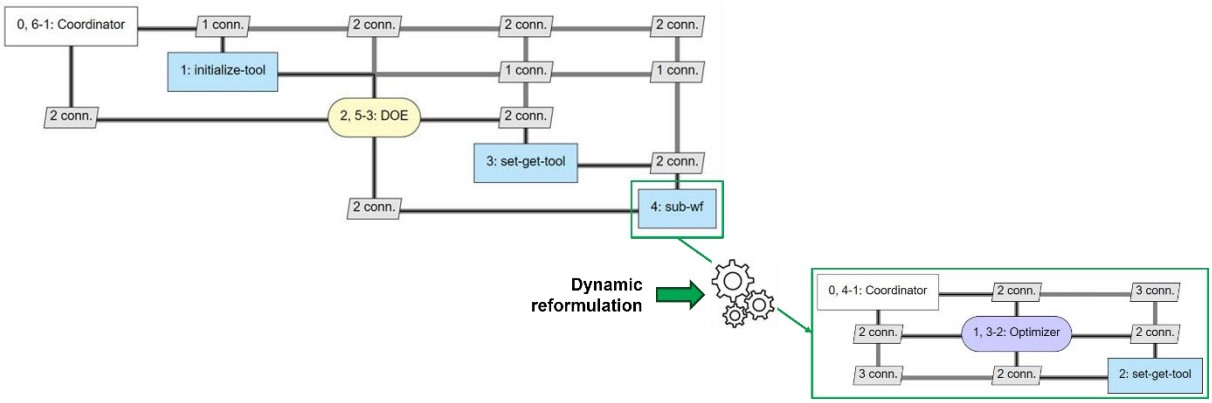


Figure 6: Main workflow featuring a nested dynamic reformulating sub-workflow

To enable the formulation of nested MDAO workflows featuring a dynamically reformulated sub workflow, as the one in Figure 6, a new variable type called 'variablePlaceholder' was introduced. These 'variablePlaceholder's (previously addressed as selectionVariable [3]) are used where the quantity and type of a variable are not known a priori. Considering the example introduced in Figure 5, Figure 7 shows the 'variablePlaceholder' variables for the 'skin_zones' and 'rib_zones' placed in both the main- and sub-workflow formulations. In the main-workflow, they are an output of the 'set_get_tool', as the tool will produce the exact quantity of the 'variablePlaceholder' variables. They are also placed in the sub-workflow, to be replaced with the actual variables once the information becomes available by running the set-get-tool.

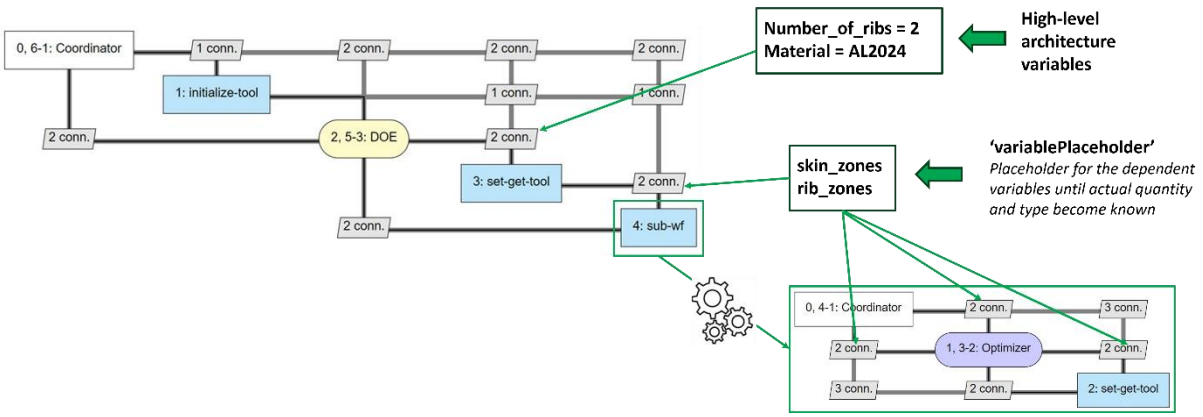


Figure 7: 'variablePlaceholder' placement in a nested dynamic reformulating workflow

In Figure 8, the process of replacing the variablePlaceholder variables with the actual variables during workflow execution is illustrated. In the main-workflow, the high-level architecture variables (Number_of_ribs and Material) are inputs to the set-get-tool tool, which is the KBE application responsible for the generation of the wing moveable models shown in Figure 5. The movable model is generated, and the quantity of dependent variables extracted. A dynamic reformulation of the sub-workflow now replaces the variablePlaceholder skin_zones and rib_zones, with the actual variables and completes the formulation of the sub-workflow. It should be noted that the number of skin_zone variables is not known a priori and is the results of the geometry operations

performed by the KBE tool on the base of the input number of ribs (and their orientation, for example).

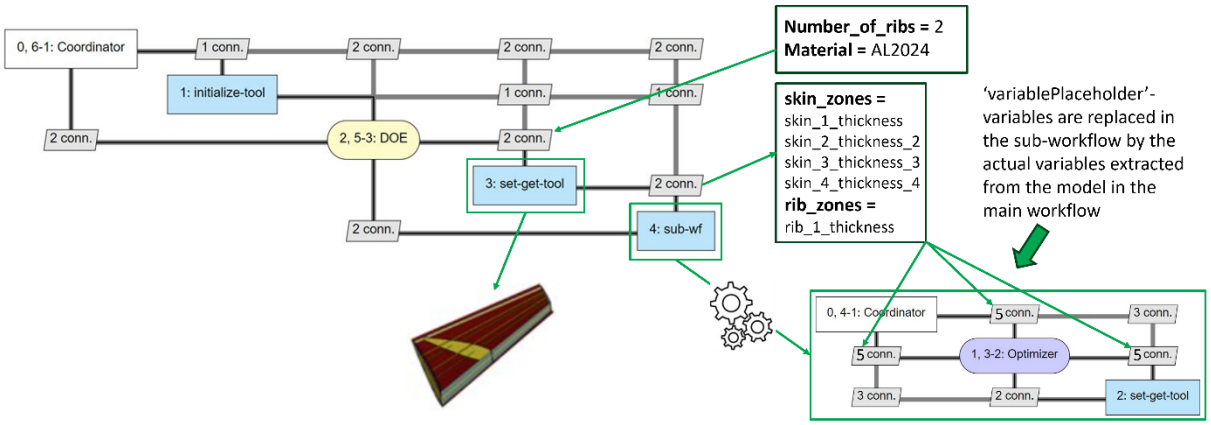


Figure 8: During workflow execution, the variablePlaceholder variables in the sub-workflow are replaced by the actual variables extracted from the generated model.

The formulation of dynamic reformulating workflows is not limited to 2 levels, in Figure 9 an example of multi-level design study is shown.

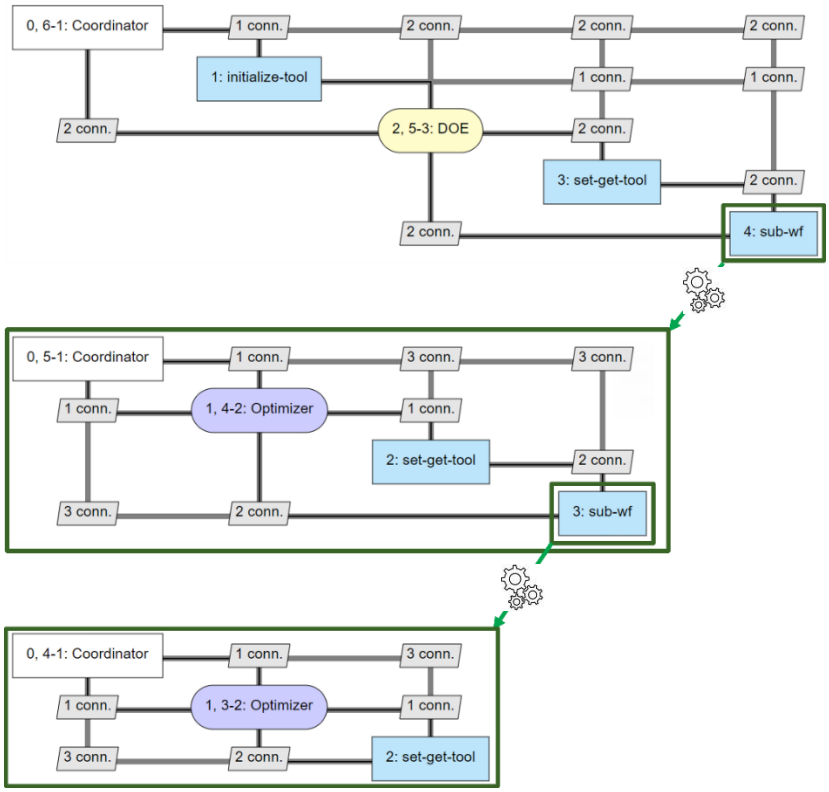


Figure 9: Example of 3-level dynamically reformulating workflow, the top level workflow performs a DOE, containing an optimization sub workflow, which in turn features another optimization sub workflow.

In the DEFAINE project, the architecture design exploration and optimization approach has been implemented for a GKN Fokker aileron design use-case [6]. The process applied in this use-case is illustrated in Figure 10, KADMOS is used in 2 instances, explained below:

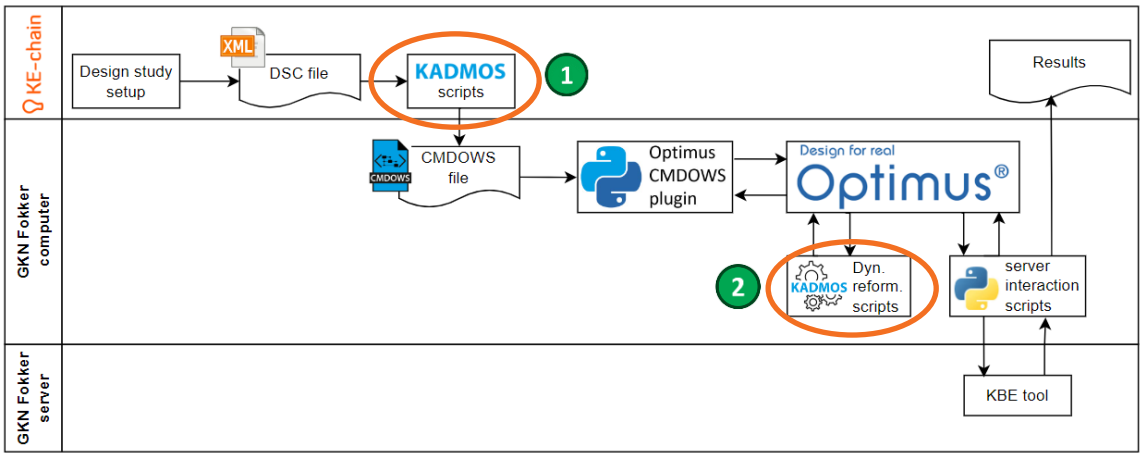


Figure 10: Process and technologies GKN Fokker moveable demonstrator, KADMOS scripts highlighted.

1. Based on a design study configured using the DSC format for configuring multi-step hierarchical design studies (see D4.1.2 [3]) a series of nested workflows is formulated using KADMOS scripts, as described in the previous section. To achieve this, KADMOS' new 'subWorkflow' feature is used as discussed in section 2.1.
2. In this implementation the workflows are executed in the PIDO tool Optimus, an example of a reformulating workflow in Optimus is show in Figure 11. During execution KADMOS scripts are called to perform the dynamic reformulation. In this process an incomplete CMDOWS formulation of the sub-workflow (meaning variablePlaceholders are present) is reformulated using KADMOS to include the now known variables to produce a completed sub-workflow formulation. The next step is materialization using the Optimus CMDOWS plugin, for more details please consult deliverable D4.2.1 [8].

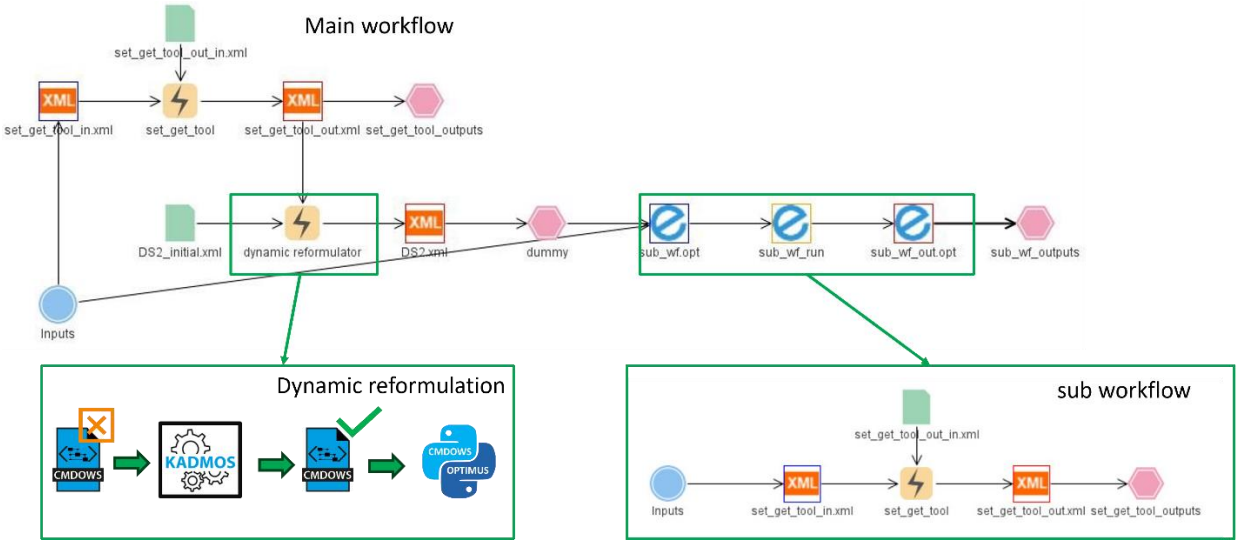


Figure 11: Example of an Optimus workflow including a dynamic reformulator script and a sub-workflow. The sub-workflow is implemented using the Opt-In-Opt function

3. Conclusions

This deliverable discusses the final release of the workflow (re-)formulation tool KADMOS. Together with the two previous versions of this deliverable, all developments of KADMOS in the DEFAINE project have been documented. In this deliverable, developments to support dynamically reformulating workflows are presented.

KADMOS is extended with the ability to formulate workflows featuring sub-workflows. Two implementations are supported. A switch that allows for the switching between multiple branches and a dynamically reformulating sub-workflow.

The extended version of KADMOS and CMDOWS and the plugin for automatic materialization of Opimus workflows have been successfully applied to the aircraft moveable demonstrator described in [6].

An extension of the XDSM, the de-facto standard visualization tool for MDAO systems, has been proposed in this work to render dynamic workflows.

These developments can be used by the industrial use cases (WP2) described in D2.1.1. [11] for the setup and generation of executable (dynamic) simulation workflows.

As this deliverable is of the type “software”, this document aims to summarise the developments and give some context for future users and developers of this methodology.