A Fully Automated Chain from MDAO Problem Formulation to Workflow Execution

van Gent, Imco; Lombardi, R.; La Rocca, Gianfranco; d'Ippolito, R

Publication date
2017

Document Version
Final published version

Published in
EUROGEN 2017 September 13-15, 2017, Madrid, Spain

Citation (APA)
van Gent, I., Lombardi, R., La Rocca, G., & d'Ippolito, R. (2017). A Fully Automated Chain from MDAO Problem Formulation to Workflow Execution. In *EUROGEN 2017 September 13-15, 2017, Madrid, Spain*

A Fully Automated Chain from MDAO Problem Formulation to Workflow Execution

Imco van Gent
Delft University of Technology, Faculty of Aerospace Engineering
Kluyverweg 1, 2629HS, Delft, The Netherlands
i.vangent@tudelft.nl

Riccardo Lombardi
NOESIS Solutions N.V.
Gaston Geenslaan 11, B4, 3001 Leuven, Belgium
riccardo.lombardi@noesissolutions.com

Gianfranco La Rocca
Delft University of Technology, Faculty of Aerospace Engineering
Kluyverweg 1, 2629HS, Delft, The Netherlands
g.larocca@tudelft.nl

Roberto d'Ippolito
NOESIS Solutions N.V.
Gaston Geenslaan 11, B4, 3001 Leuven, Belgium
roberto.dippolito@noesissolutions.com

Abstract

In this paper, a methodology to connect the multidisciplinary design analysis and optimization (MDAO) problem formulation tool KADMOS and the commercial Process Integration and Design Optimization (PIDO) platform Optimus is presented. This capability has been developed in the context of the EU project AGILE. The aim of this development is to create a combined environment that gives the MDAO design team the ability to define and formalize an MDAO problem and directly execute it with ease, without the need of the otherwise needed manual operations typically required to define the workflow in the PIDO system. The combination of problem formulation and PIDO platform execution have been tested on a small analytical MDAO problem to demonstrate its viability. Furthermore, a realistic aerostructural MDAO system of industrial relevance was also used to demonstrate the scalability of the approach for a bigger and more complex MDAO system. Results indicate that a fully automated chain is indeed possible which will make it easier for design teams to define, execute and compare different MDAO problem definitions and architectures in the time usually necessary to implement one MDAO system. Future work will focus on extending the proven capabilities of the automated chain to a wider variety of design problems and MDAO architectures.

Keywords: *MDO, MDAO, Design automation, PIDO, Systems Engineering, KADMOS, CMDOWS, Optimus*

1 Introduction

Multidisciplinary design analysis and optimization (MDAO) has been a promising method for solving complex design problems for more than three decades. Despite its potential, MDAO is not widely used in the industrial environment, which can be explained by the many hurdles that come with using this alternative design method.[1–3] Performing design using MDAO requires a different design process than any legacy design methods.[4] This shift in paradigm is the topic of the research project AGILE[a] (Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts), where the goal is to reduce key hurdles that prevent designers from switching to an MDAO design strategy.

In this paper the focus is on a specific element of the AGILE paradigm: the automated chain to go from the design problem formulation to an executable MDAO

[a]`http://www.agile-project.eu/`

workflow in a process integration and design optimization (PIDO) platform. The goal of this automation is to reduce the time and effort required to define MDAO problems and their solution strategy. In many previous collaborative projects involving MDAO it has been found that the majority of the effort (60-80%)[5] was spent on the development of the first design iteration,[4] as this requires all the design tools to be coupled in one multidisciplinary system, such that an optimizer can perform its task on that system.

The collection of steps leading to the complete integrated simulation workflow is called the MDAO development process. The full process discussed in this paper is shown in Fig. 2. This process is actually a subset of the AGILE project developments presented by Gent et al.,[6] where the implementation presented in this paper are a combination of two software platforms. The steps I-IV in this figure are performed using the KADMOS[7] (Knowledge- and graph-based Agile Design for Multidisciplinary Optimization System) package under development at Delft University of Technology (DUT). The steps V and VI are performed by means of Optimus, the commercial PIDO platform by Noesis. The development process combines the strength of KADMOS in automating and supporting the formulation of MDAO problems, while exploiting the capabilities of Optimus in executing and post-processing optimization runs. The link between the two systems is enabled by using a standardized file format called CMDOWS (Common MDO Workflow Schema).[8] The different process steps are discussed in the sections below.

2 Methodology part 1: Problem formulation and automated solution strategy

KADMOS[7] is a knowledge- and graph-based[9] system that allows the user to perform three basic actions while formulating any MDAO problem at hand:

- Formalize: An MDAO system can be created by defining a database containing all the different analysis tools and their input/output relations to a central data schema. Using these relations an MDAO problem can be formulated.
- Debug: The database is translated to a directed graph, which can be used to debug and inspect the analysis tools that have been defined.
- Manipulate: Once a complete MDAO problem has been defined using a KADMOS graph, graph manipulation algorithms can be used to impose an MDAO architecture (e.g. DOE, MDF, IDF) around it.

The different actions performed in KADMOS are further explained by going through the first four steps in Fig. 2. These steps are briefly described in the next subsections, but for a more elaborate description of KADMOS the reader is referred to Van Gent et al.[7]

2.1 *Step I: Import tools and connections*

One of the key assumptions used by KADMOS is that tool integration is performed using the central data model approach. This means that all the tools are executed by receiving their inputs based on a file that is structured according to an agreed schema. Similarly, all outputs of the tools should be written to nodes in that same schema. Thanks to the central data model, all the analysis tools are connected indirectly through the data schema, as is visualized in Fig. 1. The indirect connection between analysis tools through the schema means that a directed graph can be created to represent the full repository of tools. This graph is called the Repository Connectivity Graph (RCG), see the KADMOS graph column in Fig. 2. In the RCG every analysis tools is represented by a separate node, as is every element from the data schema. Since the elements of the data schema can be connected to different analysis tools (either as input or output), the RCG will represent the full repository of interconnected tools.

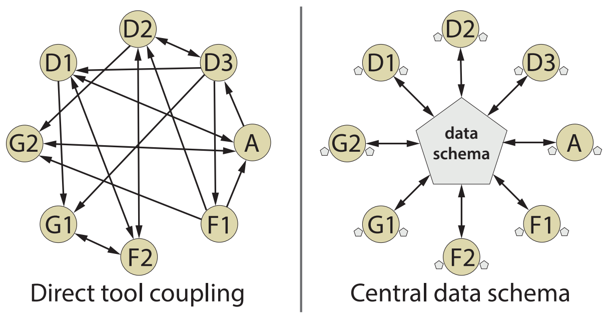
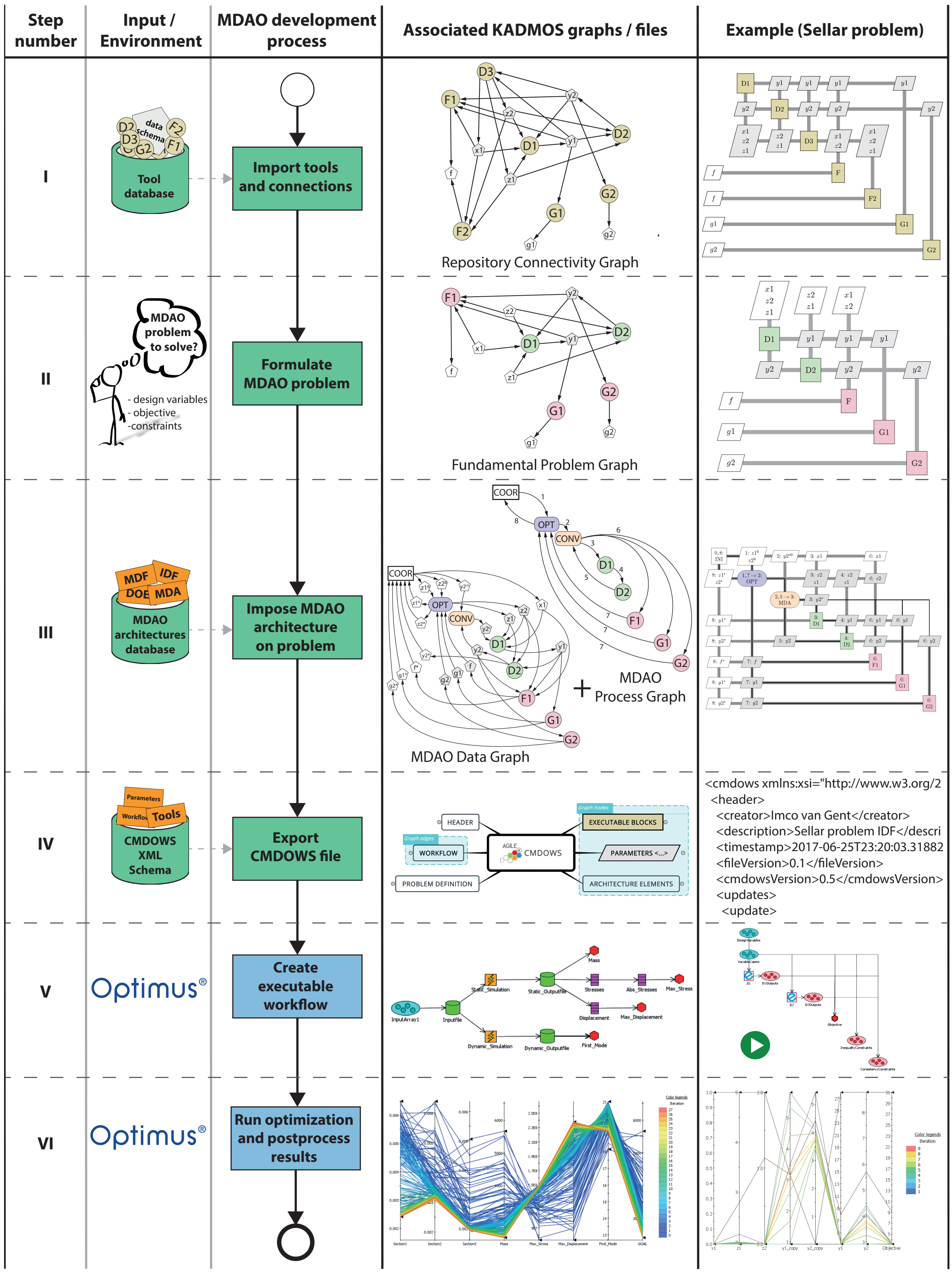


Figure 1: Visualization of tool coupling using the central data schema approach (right) vs. direct tool coupling without a schema (left).

This RCG can be visualized by using an N2 chart, as is shown in the last column of Fig. 2. This N2 chart shows through which elements of the data schema the analysis tools are connected, hence which are the coupling variables between tools. Furthermore, from the RCG one can also easily determine which data schema elements are actually inputs and outputs w.r.t. the full system. System inputs are nodes in the graph that only have outgoing edges, hence none of the analysis tools has this element as an output. Similarly, system outputs are nodes that only have incoming edges, hence none of the analysis tools are using this element as an input.

Using the RCG the MDAO system can be inspected up to the finest detail. For example, to assess whether the expected tool couplings are present in the system, whether required outputs are part of the current system (e.g. the calculation of the objective value), and whether the system inputs actually contain elements from the central data schema that can be provided as constants. Once the repository connectivity is validated, the MDAO problem at hand can be defined in the next step.



```
<cmdows xmlns:xsi="http://www.w3.org/2
  <header>
    <creator>Imco van Gent</creator>
    <description>Sellar problem IDF</descri
    <timestamp>2017-06-25T23:20:03.31882
    <fileVersion>0.1</fileVersion>
    <cmdowsVersion>0.5</cmdowsVersion>
    <updates>
      <update>
```

Figure 2: Automated MDAO development process using KADMOS (green blocks) and Optimus (blue blocks).

2.2 *Step II: Formulate MDAO problem*

Using the tools in the repository a certain MDAO problem should become solvable. However, the RCG is just a data graph which represents the connectivity between different analysis tools. The RCG first needs to be transformed into a Fundamental Problem Graph (FPG)[9] in order to be able to automatically get the full optimization strategy in step III. The FPG is a subgraph of an RCG on which three main operations have been performed:

1. Node removal
2. Node contraction
3. Node enrichment

The goals of these operations are to make the FPG as small as possible (by node removal and contraction) and to add the required additional information for the graph manipulation algorithm that will wrap the MDAO architecture around the FPG (by node enrichment). Node removal is straightforward. It simply means that any nodes that are not required to solve the problem will be removed from the RCG. If a complete analysis tool is not used, then the node representing that tool (and its connections to the data schema) can be removed. If certain system outputs are irrelevant for the problem at hand, then these data schema nodes can also be removed.

Node contraction is performed between analysis tool nodes. This is done to decrease the size of the graph. There can be different reasons to contract nodes. One of the most common reasons for contraction would be to combine analysis tools that will always be run in a certain sequence (and do not have any feedback coupling between them) in order to reduce the size of the graph. Similarly, contraction can also be performed for combining analysis tools executed in parallel.

Finally, node enrichment includes operations that will provide the details of the fundamental problem to be solved. As indicated in step II in Fig. 2, this means that the special nodes of the graph have to be marked. For example, nodes representing elements from the data schema that should be considered design variables should be marked as such. And for each design variable additional information should be added to the node, such as the bounds, nominal value, and the type of variable (i.e. real or integer). Similar operations should be performed for constraint variables and the objective of the optimization problem. Once the FPG is complete, the graph algorithms in KADMOS can be used to get the solution strategy.

2.3 *Step III: Impose MDAO architecture on problem*

Though the FPG contains the essential elements that define an MDAO problem, i.e., disciplinary tools, objectives, constraints and design variables, it does not include any strategy specification (architecture) to solve the given MDAO problem. It is in this step that an MDAO architecture[10] is imposed on the FPG. This means that additional nodes and connections are added to the graphs. The supplementary nodes would be execution blocks such as the optimizer, converger, and consistency constraint calculations, but also additional variables such as initial guesses of the design variables, copy variables used in convergers, etc. Once the imposition is complete, two graphs are created by KADMOS: the MDAO Data Graph (MDG) and the MDAO Process Graph (MPG), see row III of Fig. 2. The MDG contains the manipulated data connections between all executable blocks, including newly added nodes and their connections. The MPG stores the process of running the optimization, as it contains process step numbers, process lines, and iteration loops.

This step is the final graph manipulation performed by KADMOS, which is finally visualized by means of the XDSM (Extended Design Structure Matrix) notation,[11] as is shown in the last column of step III in Fig. 2.

2.4 *Step IV: Export CMDOWS file*

The two graphs created by KADMOS in step III (MDG and MPG) are Python graph objects, which, eventually, would need to be translated to the workflow definition supported by the PIDO tool of interest. In order to enable such a translation and allow KADMOS to operate with different PIDO tools, a neutral format to store formalized MDAO systems is being developed within AGILE. This is the so called Common MDO Workflow Schema (CMDOWS)[8] which is proposed as a new standard format to store and exchange the definition of any MDAO system between PIDO tools. CMDOWS makes use of an XML-based schema, whose top-level is shown in step IV of Fig. 2. More details on CMDOWS can also be found on the open-source repository[b]. Step IV is the last one concerning the formalization of the MDAO system. The CMDOWS file generated in this step is the input to the integration and execution part (STEP V and VI) of the overall MDAO process illustrated in Fig. 2.

3 Methodology part 2: Workflow materialization

Optimus is a process integration and design optimization system environment developed by Noesis Solutions.[12] The software environment provides the means to integrate different processes within a single, connected, simulation workflow, automate the tasks required to execute these processes, explore the design space and perform the design optimization using one of the many included optimization algorithms or even by adding a custom made one. The integration of different processes in a unique framework requires the definition of how a set of input (or design) variables is linked to a set of output variables. In Optimus, this is usually done by means of a Graphical User Interface (GUI), Fig. 3, that provides the tools required to simulate the design process. All these entities are then linked together to form an oriented graph (i.e. the simulation workflow) that allows defining in a logical and quantitative way

[b]`http://cmdows-repo.agile-project.eu`

how the data is transferred from the input (cyan icons) to the output variables (red marks) passing through a series of processing actions (yellow blocks).

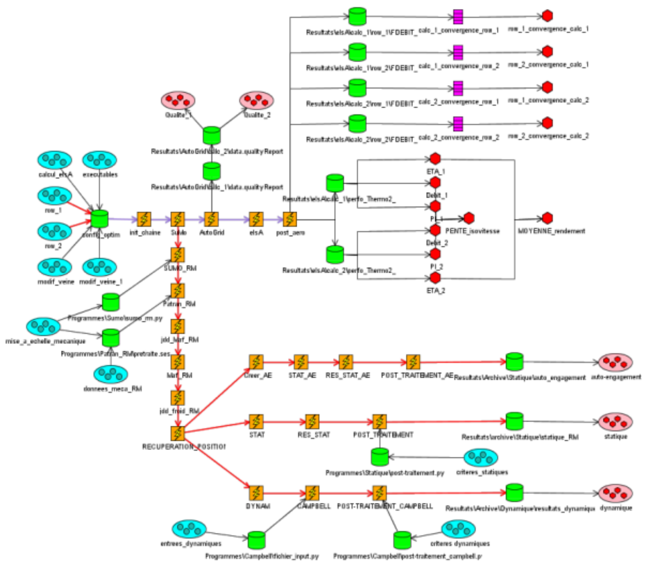


Figure 3: Simulation workflow implemented using Optimus.

In Optimus simulation workflows can be associated to one or more analysis methods, i.e. algorithms integrated in the program kernel that can be used to investigate the design space in order to explore the model behavior, extract the most significant model features and build surrogate models, perform uncertainty propagation and sensitivity analysis, and execute single and multiple outputs design optimization.

Instead of using the GUI to build simulation workflows, they can also be created using a development environment that exposes the GUI commands as Python functions, thereby making both the automatic generation of standardized workflows and the execution of repetitive tasks possible. The construction of specialized and complicated workflows with high number of variables, constraints and disciplines is also possible through the Python-based development environment. However, due to time and implementation limits, this approach is rarely adopted. Furthermore, such code-based system definitions have both poor maintainability and limited room for customization without further increase in the code complexity.

The availability of the problem formulation system KADMOS and the standardized workflow storage format CMDOWS, allow the creation of a set of routines that automatically convert the formalized workflow (created by KADMOS and stored in a CMDOWS file) into an actual workflow ready to be executed, independent of the complexity of the workflow and the number of involved variables. The CMDOWS file is the only information exchanged between KADMOS and Optimus. Its XML-based format is parsed into a Python dictionary that converts the completely generalized implementation into a more Optimus-oriented formalization of the graph. This includes the identification of:

- Design variables, along with their name, id, type (float, integer, string) and range.
- Objective functions (formula and involved variables).
- Constraints (both boundary values and equations as well as the name of the constrained variables).
- Separation between input, output and coupling variables.
- Components (also called disciplines).
- Disciplines execution order.

Equations are identified and converted in order to ensure that they will be correctly interpreted by Optimus (by adjusting operators and functions name). The information summarized in the dictionary are subsequently refined and checked before the final materialization of the Optimus project. The construction of the MDAO workflow, also referred to as materialization, is divided into a sequence of steps that have to be performed in the assigned order as they incrementally enrich the information in the workflow itself. For example design variables have to be defined before disciplines. In fact, the materialization process mimics the operations that would have been performed by a user through the conventional GUI-based approach, to create the same MDAO using the XDSM as reference.

The sequence of operations can be broadly divided into five main steps, shortly described hereafter. The first four are completely automated and end with the creation of a fully functional simulation workflow; the last step requires a minimum user effort to define the detailed settings, for example the optimization algorithm to be used and its settings like tolerances, maximum iterations, etc.

The CMDOWS file, although mainly aimed at the definition of the MDAO problem, can be used to infer a lot of details about the disciplines as well; specifically all the input and output variables and parameters are listed. Exploiting this knowledge, is possible to create synthetic workflow templates that allow for the integration of the disciplinary tools with reduced effort. To this end, the materialization has been implemented based on structure consisting of three levels, Fig. 4. The objective is to maintain a formal separation between the top (MDAO workflow) level and the bottom (disciplinary workflow) level without hindering the information exchange between them. The intermediate level depicted in the left part of Fig. 4 is used to connect the top level to a remote disciplinary workflow. In this case, the information exchange occurs by files through an FTP file transfer protocol. Alternatively, the middle level can also be skipped when the disciplinary workflow is executed locally.

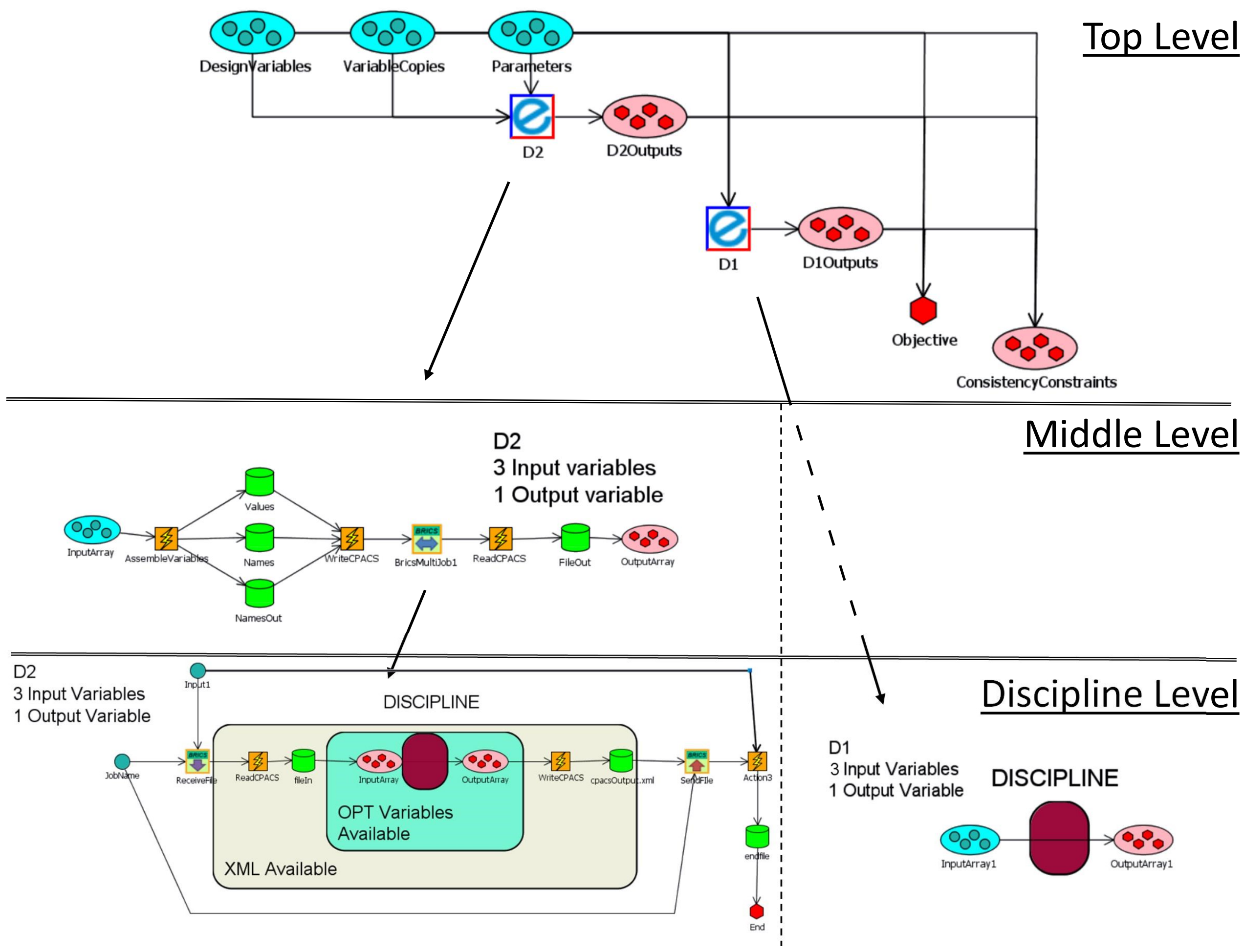


Figure 4: Structure of the materialized workflows, highlighting the distinction between top, middle and discipline levels.

3.1 *Check of discipline variables name*

As mentioned, the variable definition is the first step of the process. Their properties, such as name, type and range are described in CMDOWS. Discipline, Middle and Top level variable names do not have to match; i.e. a discipline can have a variable named 'mass' that is corresponds to 'total fuel' in the Top level. This mapping information is contained and extracted from the CMDOWS file. Aside from those performed by KADMOS, an additional check is executed to ensure that every elements have unique variable names. As every Optimus workflow is a representation of an ordered graph, the creation of nodes with the same name is not allowed, in order to prevent possible conflicts and ambiguities. If needed, a naming convention is enforced to solve the conflicts. Subsequently a coherence check is executed to ensure the right connection of the local variables and the global ones.

3.2 *Generation of the disciplinary workflow*

The disciplines in CMDOWS are depicted in terms of inputs and outputs whereas the relationship that binds them is not reported. This is on purpose as the main aim of KADMOS is the description of the MDAO. Consequently the generated workflows can only have coherent variables ready to be interfaced, while the setup of an input-output connection is left to the final user, who is free to use mathematical equations, surrogate models, or analysis tools (such as Computational fluid dynamic, structural Finite Element solvers or parameterized CAD Models). In this sense the workflow has been intended as a generic template that could be used to wrap a dedicated analysis component.

The two types of discipline materialization have been implemented, Fig. 4. In the simplest scenario (right), the discipline is available on the same platform that hosts the MDAO; in this case a direct connection can be created with the Top level that links its variables to their discipline counterparts. The data flow is managed by the Optimus kernel. Remote disciplines involve an intermediate level to provide a means to control the information exchange. This Middle level creation is completely automated and maps all the variables inherited from the MDAO to an XML-type file. The file is then forwarded to the lower level where the inputs are extracted and made available to be interfaced with any analysis tool. Outputs are treated likewise, added to the XML file and returned to the middle level.

3.3 Definition of the variables

The construction of the top-level MDAO simulation workflow starts with the definition of the variables. In the simplest optimization problem, only design variables and objective are required. During the set up of a multidisciplinary design optimization, the choice of the architecture defines how disciplines are linked. The subsequent adjustments on the structure of the problem may involve the introduction of 'auxiliary' variables needed to ensure the information exchange between the disciplines such as coupling variables, coupling variable copies and, related to the latter, consistency variables (to ensure that the optimum design is a consistent solution). Both the architecture definition, the distinction between the variable types and their connections are stored in CMDOWS, categorized as follows:

- Design variables, variables in the MDAO problem that are always under the explicit control of an optimizer/user. Upper and lower boundaries have to be specified as they define the design space.
- Design objective, the quantity that is influenced by the design variables and has to be minimized/maximized.
- Coupling variables, used to exchange information among sequential disciplines.
- Constraints, boundaries on the objectives (specified quantitatively).

Design and consistency constraints are introduced in the workflow as additional, bounded variables.

3.4 Generation of the disciplines in the MDAO system

The workflows created during the second step are linked to the Top level using an Optimus-in-Optimus construct,[12] which is a standardized interface that allows for the connection of simulation blocks on multiple levels by specifying the inputs that have to be transmitted and the outputs that must be retrieved. More in detail, the link requires the discipline to be wrapped in a tasklist, (one or more analysis methods defined and associated to a "frozen" workflow that will be executed sequentially); this dictates that disciplines have to be created before the finalization of the MDAO level. The Top level inputs are associated to the lower workflow design variables value and range. The extracted variables can be the result of any method, including optimization, thus making multiple level optimization possible. This step, especially for high number of parameters to be connected is time consuming and a recurrent source of errors for the users. The automation solves both issues by exploiting the variable map constructed by KADMOS.

3.5 Additional Components

The assembled Top level has all the elements inherited from CMDOWS and is ready to be executed. The run step requires the association of the simulation workflow with one or more analysis methods. They may be a simple nominal run (single execution with specified values of the design variables), a design of experiments (multiple executions to explore the design space) or an optimization. The definition of the analysis method is not covered by the CMDOWS formulation as the specific application does not depend on the architecture of the system; the same workflow can be reused multiple times for different activities, so their specific implementation, like the choice of the optimization algorithm to be used or its parameters (namely population size, convergence tolerance) are left to the user. The Top level workflow can be considered a discipline as well, thereby making it possible to connect it to a higher level MDAO design simulation to create projects with increasingly larger complexity.

4 Materialized simulation workflow

The created Optimus workflow resembles an XDSM diagram (see examples in Fig. 8 and 7 respectively). The specific position of the disciplines in Optimus is not relevant as long as the connection sequence is correct. In other terms, the execution sequence is driven only by the ordering of oriented graph. However, to improve the readability of the workflow, positioning rules similar to those implemented in XDSM have been implemented. Consequently, the final results shows similarity with N2 type diagrams, where the disciplines are positioned on the 'main diagonal', design variable on the top left corner and objectives in the lower right. The definition of the optimization algorithm and the initial values are not depicted in the graph. This is due to the aforementioned separation that is made in Optimus between the workflow, and the analysis method attached to it.

It is worth noting that for the test cases shown in this work the simulation workflows at the discipline level only exploit a limited subset of the features available in Optimus, specifically input and output (variables and constraints) definition, discipline hierarchy construction, sub-workflow link connection and iteration of loop cycles. These modules allow for the implementation of a general purpose MDAO that only uses Optimus native variables and links.

In a more general application, the definition of the disciplinary workflows might require dedicated integration steps to be performed. As an example, many analysis tools are not able to receive inputs directly and rely on files with a standardized format; others require specific commands to be executed or parameters that depend on the characteristics of the specific problem. For several software, dedicated interfaces have been developed and are shipped with the integration platform, however, with the central data schema approach any disciplinary tool will always have to be linked to the central data schema, meaning that the Top level workflow is independent of the complexity of the disciplinary workflows. Therefore, KADMOS and CMDOWS do not consider the complexity of the disciplinary workflow and handle them as black boxes based on the input/output relations with the data schema.

As mentioned before, the selection of the optimization algorithm is a final manual step. This selection is not a trivial process; local vs. global algorithm, selection of the starting point, or method settings can make a huge difference in the solution of high-dimensional, non-linear problems. However, the relation between the selection of the optimization algorithm and the MDAO architecture was outside the scope of this work and is left for future developments. At this stage, general purpose optimization algorithms have been used to demonstrate the KADMOS-Optimus interoperability.

5 Test case: Sellar problem

The Sellar problem[13] is a small mathematical MDAO problem that has been extensively used in literature to test new MDAO developments. More details on the Sellar problem and KADMOS can be found in earlier work.[7] The last two columns of Fig. 2 depict the implementation of the Sellar problem in KADMOS. In step III, KADMOS is able to automatically impose different optimization strategies around the same problem. In this test case, the MDF (Multidisciplinary Feasible) and the IDF (Individual Discipline Feasible) approaches have been automatically imposed on the problem. As a final step KADMOS stores these optimization strategies in CMDOWS files for materialization in Optimus.

The materialization procedure has been tested on the Sellar problem with these two different architectures to test both the capacity to correctly assemble them, as well as to execute them and verify their exactness. The discipline workflows have been manually modified to express the output from the inputs using the canonical formulas.

5.1 Independent Design Feasible

In the IDF decomposition two state variables copies are added to the design variables for a total of five dimensions and two consistency constraints.

The Optimus materialization of the problem is reported in Fig. 5. The initial values for the design variables x1, z1 and z2 were 1.0, 5.0 and 2.0; the two state variables y1_copy and y2_copy have been initialized at 3.16 and 0.1 respectively. A sequential quadratic programming (SQP) optimization algorithm has been selected to test the workflow. SQP is a general purpose algorithms for solving smooth nonlinear optimization problems, at least under assumptions that the problem is not too large, the functions and gradients can be evaluated with sufficiently high precision and the problem is smooth and well-scaled. The stop

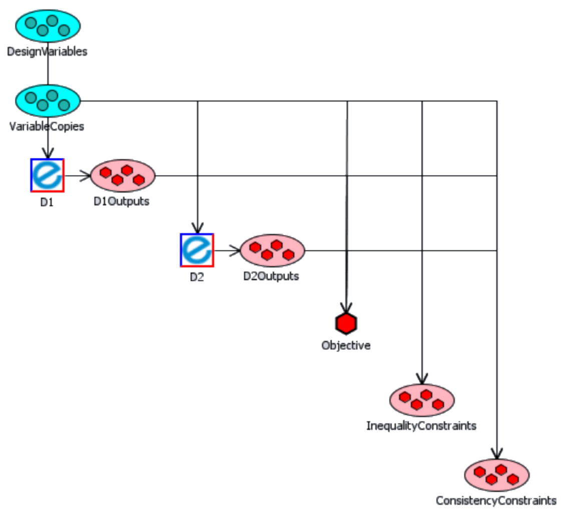


Figure 5: Materialized Sellar problem, IDF Architecture.

condition has been reached in nine iterations, Fig. 6

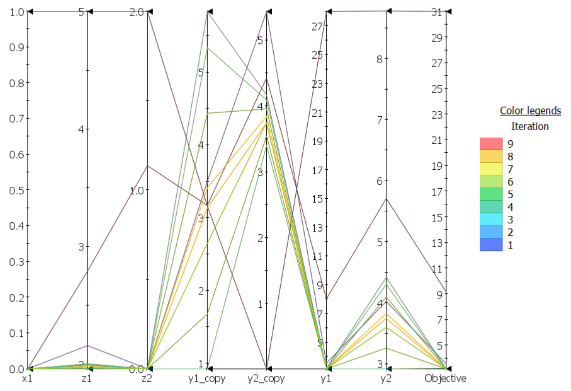


Figure 6: Parallel coordinates chart for SQP on IDF Architecture.

The achieved solution, 0., 1.97739, 0. leads to an objective value (3.18347) in line with the theoretical one.

5.2 Multidisciplinary Design Feasible

The MDF formulation of the problem (Fig. 8) using the same starting conditions and using the same optimization method (SQP) stopped after a lower number of iterations (5) but took significantly longer time to execute due to the internal convergence loop. The solution is less accurate 0., 2.36, 0. with an objective function value of 3.176. This has been caused by the premature end of the convergence loop, due to reached limit number of executions instead of convergence condition satisfaction. Apparently, performing the simulation with the default settings was inadequate for the specific test-case, but finding the right settings and solving the optimization problem was out of scope of the present work.

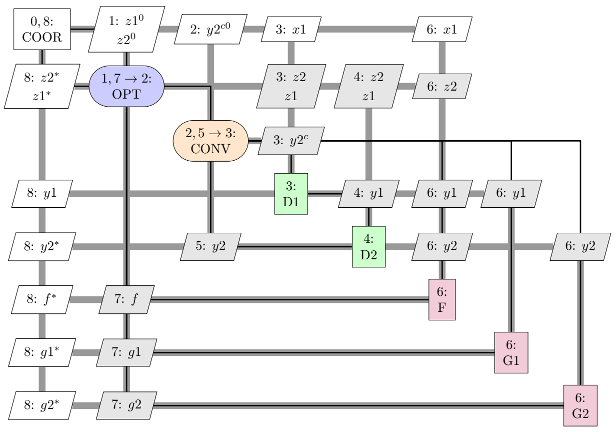


Figure 7: XDSM for the Sellar problem, MDF Gauss–Seidel Architecture.

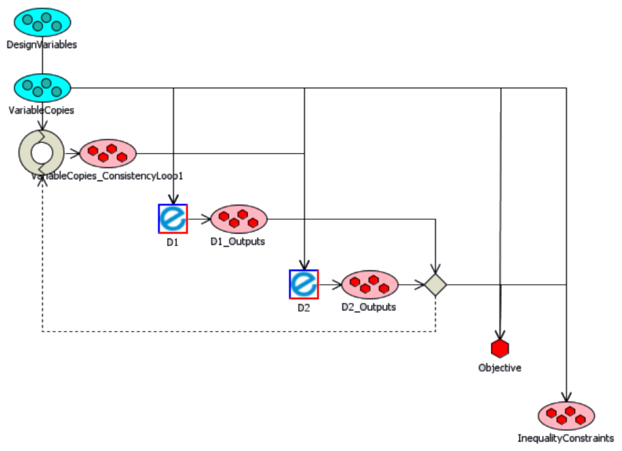


Figure 8: Materialized Sellar problem, MDF Architecture.

6 Test case: Aerostructural wing design optimization

The Sellar problem test case that was presented in the previous section constituted a proof-of-concept which is based on a small MDAO system of simple analytical functions and a low amount of variables. The automated MDAO development process depicted in Fig. 2 would only be valuable if it is able to provide the same automation for realistic, larger MDAO systems with a much higher level of complexity. This is the goal of this second test case: an aerostructural wing design optimization. This test case uses a collection of wing design tools from Technische Universiteit Delft (DUT), i.e. aerodynamic solver, weight estimation tool, which have been linked together using the central data schema for aircraft called CPACS[14,c] (Common Parametric Aircraft Configuration Schema).

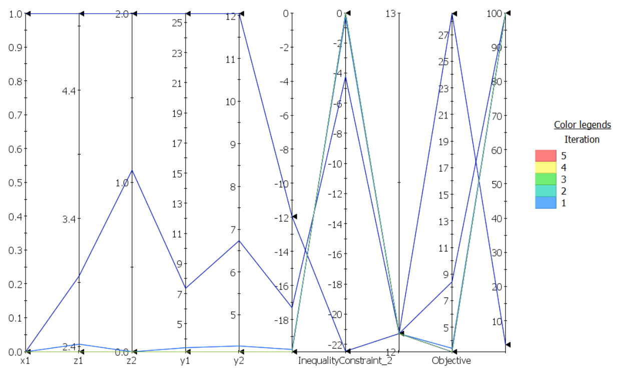


Figure 9: Parallel coordinates chart for SQP on MDF Architecture.

Note that wing design case is also discussed by Van Gent et al.[7] where it is used to automatically create RCE[d] workflows directly from KADMOS. The test case presented here contains two additions to this case, namely the use of the CMDOWS format to store the neutral representation of the workflow created by KADMOS and the ability of Optimus to use the CMDOWS file to materialize an executable workflow, thereby demonstrating the ability to exchange information between the formulation of the MDAO system in KADMOS and the automatic execution of that formalized system.

6.1 MDAO development process

The test case is described following the MDAO development process depicted in Fig. 2.

6.1.1 Step I: Tool database and RCG

The collection of DUT tools that is available in the database is summarized in Table 1. Twelve different tools are available, however, some of the tools have multiple execution modes. Each of these modes is seen as a different function block by KADMOS when the database is imported (later, some of these modes are merged again for the creation of the FPG). After the database has been imported the RCG is the web of data describing the connectivity of the tools in the database. The amount of nodes (28196) and edges (37509) does not allow a readable way to directly visualize the graph itself, however, an XDSM data flow can be created, as is depicted in Fig. 10 for a subset of the database. In this chart only the total amount of tool connections is shown in the off-diagonal blocks and it is clear that many connections between the tools have been found.

6.1.2 Step II: Fundamental problem graph

Based on the RCG an MDAO problem can be defined in terms of the FPG. In this test case the IDF architecture will be demonstrated for the aerostructural optimization of the wing. The general optimization problem (before imposing

[c]`http://cpacs.de/`

[d]`http://rcenvironment.de/`

Table 1: Tools in the DUT tool database

Tool name	Description	Number of modes	Mode	Mode description	Used in FPG?
HANGAR	Tool loads CPACS file with aircraft geometry.	7	AGILE DC1 WP6 wing startpoint	Adjusted design based on AGILE DC1 L0 wing mode	yes
			AGILE DC1 L0 MDA	Aircraft design based on AGILE design campaign 1	no
			AGILE DC1 L0 wing	Wing design based on AGILE design campaign 1	no
			Boxwing AGILE Hangar	Boxwing aircraft example	no
			D150 AGILE Hangar	Conventional aircraft	no
			NASA CRM AGILE Hangar	NASA Common Research Model aircraft	no
			ATR72 AGILE Hangar	ATR72 aircraft model	no
INITIATOR	Tool initiates an aircraft design based on top-level aircraft requirements.	N/A	N/A	N/A	no
SCAM	Simplified CPACS Aircraft Morphing. Adjust aircraft geometry in different ways.	5	wing taper morph	Morph wing taper	yes
			wing sweep morph	Morph wing sweep	yes
			wing dihedral morph	Morph wing dihedral	yes
			wing root chord morph	Morph wing root chord	yes
			wing length morph	Morph wing length	yes
GACA	Geometrical Analysis of CPACS Aircraft components.	2	mainWingRefArea	Determination of wing reference area	yes
			mainWingFuelTankVol	Determination of the wing fuel tank volume	yes
Q3D[15]	Quasi-3D Aerodynamic solver.	3	VDE	Viscous Drag Estimation	yes
			FLC	Flight Load Case (vortex lattice method)	yes
			APM	Aeroperformance Map	no
EMWET[16]	Wing mass estimation tool.	N/A	N/A	N/A	yes
SMFA	Simplified Mission Fuel Analysis: Breguet's range equation.	N/A	N/A	N/A	yes
PHALANX[17]	Flight dynamics toolbox.	4	Full Lookup	Full = full dynamic model	no
			Full Simple	Simple = emperical engine deck	no
			Symmetric Lookup	Symmetric = only longitudinal dynamics	no
			Symmetric Simple	Lookup = external engine deck	no
PROTEUS[18]	Aeroelastic wing analysis tool.	N/A	N/A	N/A	no
MTOW	Mass calculation tool for maximum take-off weight.	N/A	N/A	N/A	yes
OBJ	Tool containing normalized objectives.	N/A	N/A	N/A	yes
CNSTRNT	Constraint value analysis.	2	wingLoading	Determination of wing loading constraint value	yes
			fuelTankVolume	Determination of fuel tank volume constraint value	yes

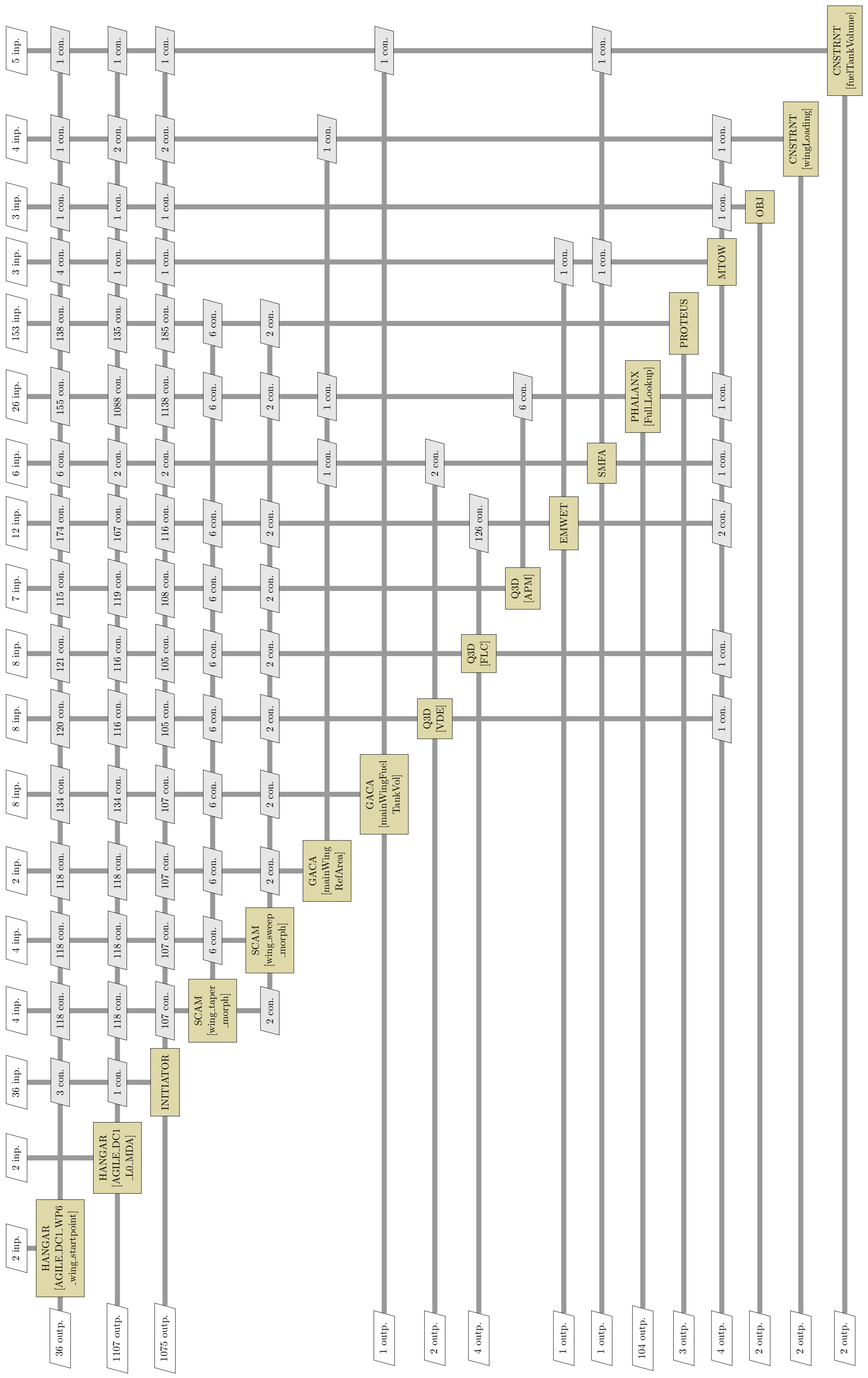


Figure 10: XDSM data flow visualization of a subset of the RCG (not all tool modes are shown to maintain readability) [con. = connection, inp. = input, outp. = output]

the MDAO architecture) is mathematically defined as:

$$\begin{aligned} &\text{minimize } MTOW \\ &\text{with respect to: } \Lambda_i, \lambda_i, \Gamma, b, c_{\text{root}} \\ &\text{where } i \text{ indicates a wing segment:} \{1,2\} \\ &\text{subject to: } \frac{W}{S} \leq \frac{W}{S}_{\text{ref}} \\ &V_{\text{fuel}} \leq V_{\text{fueltank}} \end{aligned}$$

Several operations are required to compose the FPG based on the RCG in Fig. 10. More details on these steps and the requirements for a valid and complete FPG have been reported in earlier work.[7] The basic operations required for this test case are:

1. Function block operations:
   (a) Remove unnecessary functions, such as INITIATOR, PHALANX, PROTEUS
   (b) Remove unnecessary function modes, such Q3D APM mode and several HANGAR modes
   (c) Merge functions modes into one block for SCAM, GACA, CNSTRNT
   (d) Merge functions that are to be run as a sequential subworkflow: Q3D[FLC] + EMWET, Q3D[VDE] + SMFA
2. Variable operations:
   (a) Mark design variables (e.g. sweep angles, taper ratios, wing span) and specify their bounds
   (b) Mark objective variable (normalized MTOW coming from OBJ tool)
   (c) Mark constraints (outputs of the CNSTRNT tool)
3. Graph operations:
   (a) Indicate logical analysis order of the tools
   (b) Set the required MDAO architecture (IDF in this case)

After these operations the FPG is complete. The graph is depicted using the XDSM data flow in Fig. 11.

6.1.3 Step III: Imposing the MDAO architecture

KADMOS can now fully automatically impose the IDF architecture on the FPG from step II. More details on this step are also available in earlier work.[7] The graph-based algorithms in KADMOS will create two graphs in order to provide a formal specification of the optimization strategy: the MDAO Data Graph (MDG) and MDAO Process Graph (MPG). These two graphs can be visualized in a single visualization using the XDSM, see Fig. 12. Note that the MDG and MPG are both neutral specifications of the optimization strategy, therefore they are not specific for any workflow execution software such as Optimus.

6.1.4 Step IV: Export CMDOWS file

The final operation performed by KADMOS is the export of the MDG and MPG combination to a single XML file according to the CMDOWS. This CMDOWS file is the starting point for the Optimus parser that translates the neutral KADMOS definition of the optimization strategy into an executable workflow.

6.1.5 Step V: Create executable Optimus workflow

The materialization sequence is applied to the exported CMDOWS file. The resulting Top level workflow is shown in Fig. 13. Unlike the XDSM, pre-processing disciplines as HANGAR or SCAM are not marked differently. By default all disciplines have been considered as remote tools, thus the construction of the middle level has been performed for the six tools.

6.1.6 Step VI: Optimization results

The execution of the complete MDAO has not been performed due to time constraints whereas the correctness of the Middle (specifically, data translation and transfer functionality) and Discipline (tool integration) levels has been tested for the GACA component alone.

6.2 Test case discussion

The aerostructural wing design case represents a more comprehensive and realistic test scenario than the Sellar problem, both for the formalization and the materialization sequences. Unforeseen issues have been detected, and later resolved, regarding special characters in variables and discipline names. The test case also allowed to benchmark the materialization performances in a mid-size MDAO problem, with the number of involved variables in the range of hundreds. The total run time to create the Top, Middle and Discipline level workflows with the parser is about 20 seconds. For comparison, the time required to set up manually all the connections (around 700) in the Optimus-in-Optimus would have been several hours at best. The execution of the materialized workflows is still to be performed and tested, as well as the creation of other MDAO architectures than the IDF shown in this paper.

7 Conclusions and further developments

As was presented in this paper, a new MDAO development process has been defined and implemented to demonstrate a fully automated chain from problem formulation to workflow execution. Using the Sellar problem, and to some extent the aerostructural wind design problem as demonstrators, it has been shown that KADMOS is able to define different optimization strategies (MDF, IDF) and store the strategy in a neutral format (CMDOWS) that can be imported by a PIDO platform. The Optimus PIDO platform was able to read the CMDOWS format and create a ready-to-be-executed simulation workflow.

The CMDOWS format was shown to be well-suited for the automatic creation of executable workflows, however, still some manual actions needed to be performed by the

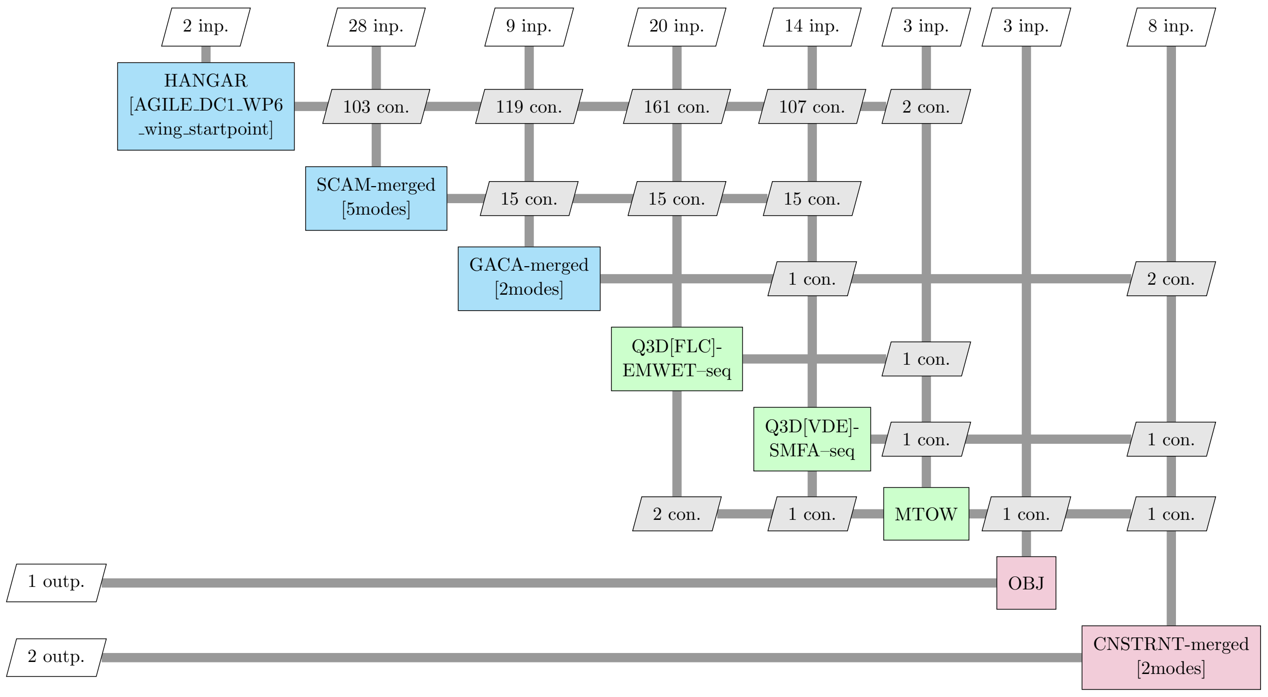


Figure 11: Visualization of the fundamental problem graph for the aerostructural wing MDAO problem. [con. = connection, inp. = input, outp. = output]

user to specify detailed workflow settings such as the optimization algorithm and its tolerances. This has shown that the CMDOWS format can be further improved by extending the schema to include such settings as well, provided that they can be represented in a neutral format and are not merely specific for a particular PIDO platform.

From the execution time perspective, the materialization of the disciplines as separated entities that are connected using task lists, has introduced a significant overhead during the evaluation phase (in the order of a few seconds per execution of a discipline). Additional overhead is introduced by the FTP file transfer required by the remote tools. This is particularly relevant on analytic problems where the execution of the disciplines itself is almost instantaneous. As the problem complexity and the requirement for flexibility increase (as during a product design phase), the advantages given by the possibility to easily reconfigure the whole MDAO in minutes instead of hours and the capacity to select the decomposition architecture that reduces the number of discipline executions become evident. Furthermore the materialization of all the discipline components in the Top level, would lead to a complex and poorly manageable/readable workflow.

The materialization of the aerostructural wing design has shown that the methodology can be applied to realistic scenarios. In this case the automatic MDAO construction removes a significant load from the final user, minimizing the number of operations that the design team has to perform in order to be able to run its simulation. Future work will focus on extending the proven capabilities of the automated chain in Fig. 2 to a wider variety of design problems and MDAO architectures.

Acknowledgments

The research presented in this paper has been performed in the framework of the AGILE project (Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts) and has received funding from the European Union Horizon 2020 Programme (H2020-MG-2014-2015) under grant agreement n° 636202. The authors are grateful to the partners of the AGILE consortium for their contribution and feedback.

References

[1] Belie, R. Non-technical barriers to multidisciplinary optimisation in the aerospace industry. In *9th AIAA/ISSMO Symposium of Multidisciplinary Analysis and Optimisation*, 4–6, (2002).

[2] Shahpar, S. Challenges to overcome for routine usage of automatic optimisation in the propulsion industry. *Aeronautical Journal* **115**(1172), 615 (2011).

[3] Simpson, T. W. and Martins, J. R. R. A. Multidisciplinary design optimization for complex engineered systems: report from a national science foundation workshop. *Journal of Mechanical Design* **133**(10), 101002 (2011).

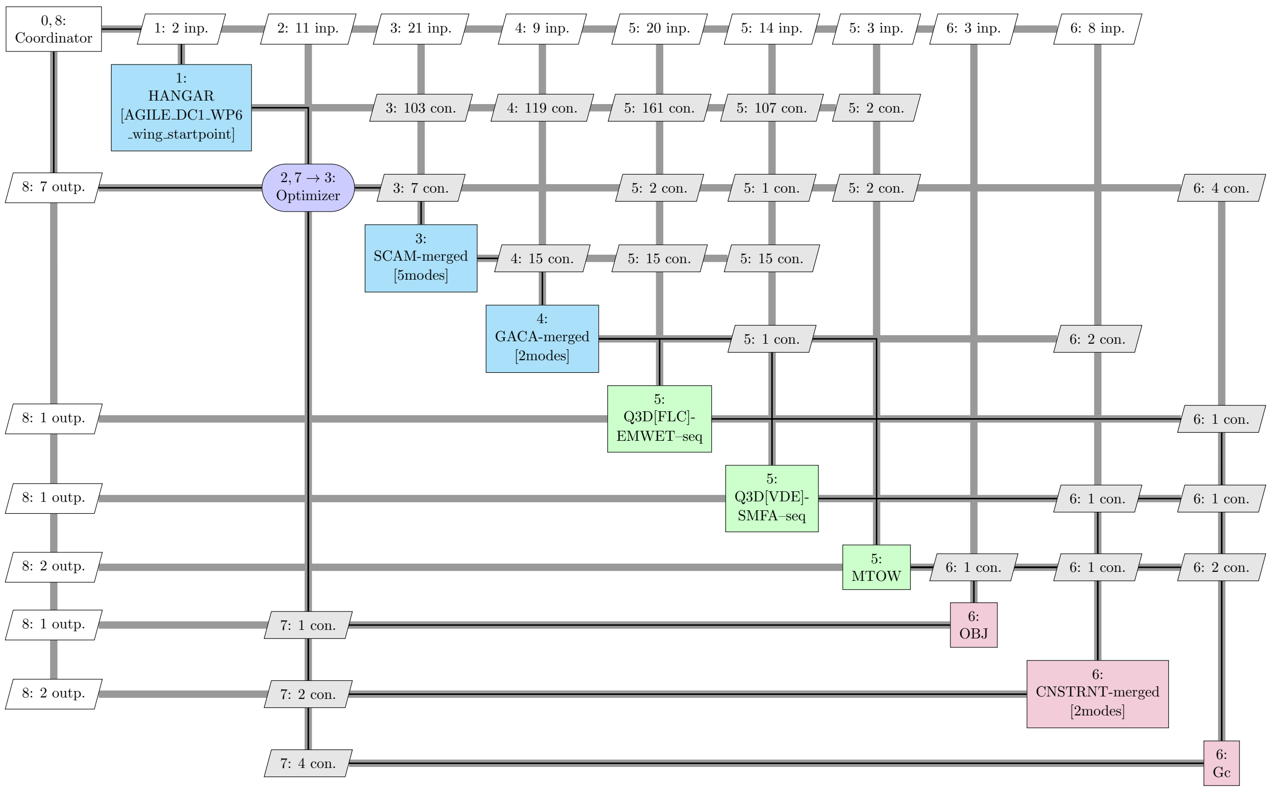


Figure 12: Formal specification of the IDF optimization strategy for the aerostructural wing design problem in an XDSM. [con. = connection, inp. = input, outp. = output]

[4] Flager, F. and Haymaker, J. A comparison of multidisciplinary design, analysis and optimization processes in the building construction and aerospace industries. In *24th international conference on information technology in construction*, 625–630, (2007).

[5] Ciampa, P. D. and Nagel, B. Towards the 3rd generation MDO collaboration environment. In *30th Congress of the International Council of the Aeronautical Sciences*, (2016).

[6] van Gent, I., Ciampa, P. D., Aigner, B., Jepsen, J., La Rocca, G., and Schut, E. J. Knowledge architecture supporting collaborative MDO in the AGILE paradigm. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, (2017).

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

[8] van Gent, I., Hoogreef, M. F. M., and La Rocca, G. CMDOWS: A Proposed New Standard to Formalize and Exchange MDO Systems. In *6th CEAS Air and Space Conference*, (2017).

[9] Pate, D. J., Gray, J., and German, B. J. A graph theoretic approach to problem formulation for multidisciplinary design analysis and optimization. *Structural and Multidisciplinary Optimization* **49**(5), 743–760 (2014).

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

[11] Lambe, A. B. and Martins, J. R. R. A. 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).

[12] Noesis Solutions, Optimus Rev 10.19 - User's Manual (available on request), (2017).

[13] Sellar, R. S., Batill, S. M., and Renaud, J. E. Response surface based, concurrent subspace optimization for multidisciplinary system design. *AIAA paper* **714**, 1996 (1996).

[14] Nagel, B., Böhnke, D., Gollnick, V., Schmollgruber, P., Rizzi, A., La Rocca, G., and Alonso, J. J. Communication in aircraft design: Can we establish a com-

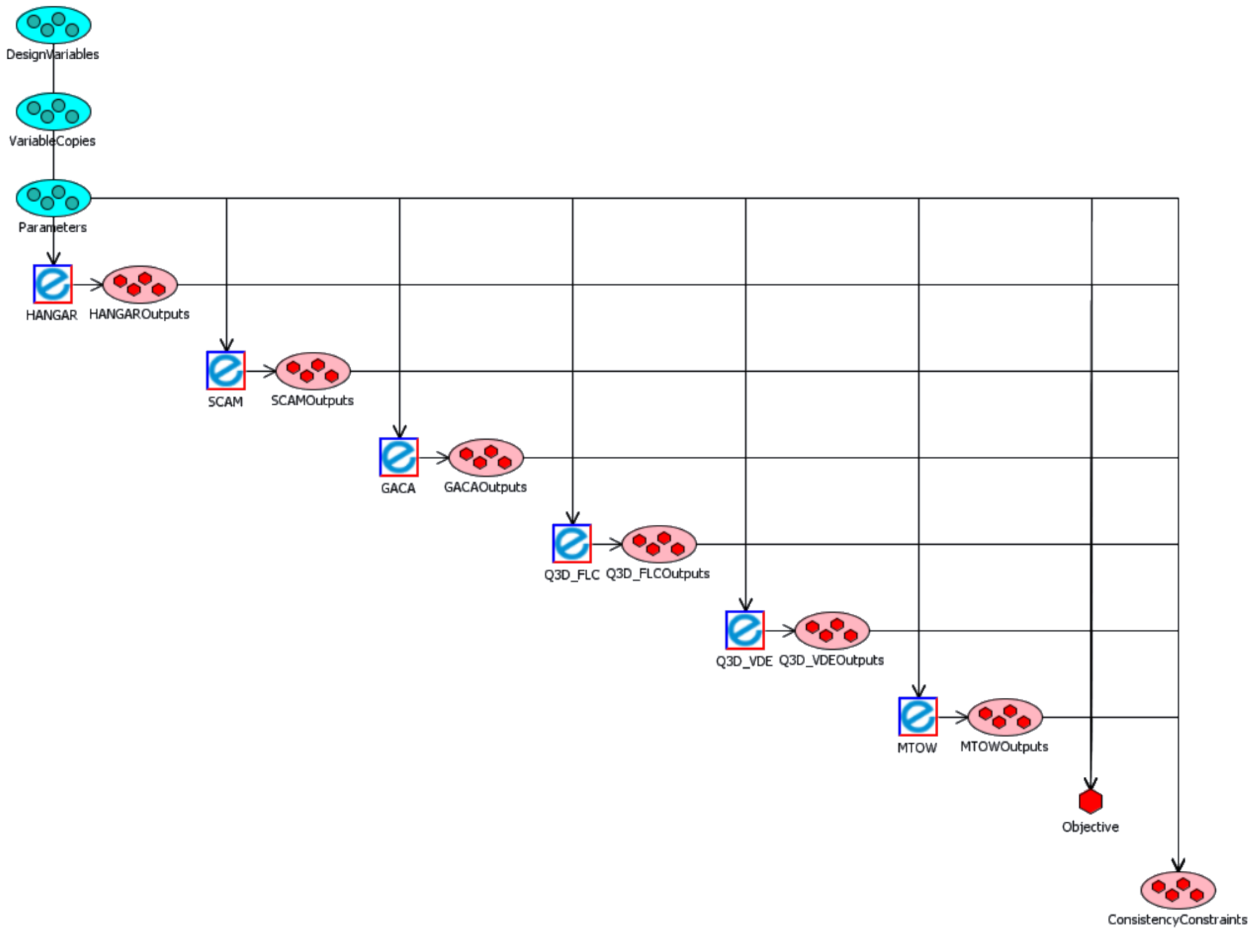


Figure 13: Materialization of the Top level workflow for the IDF wing design optimization.

mon language. In *28th International Congress Of The Aeronautical Sciences, Brisbane*, (2012).

[15] Mariens, J., Elham, A., and van Tooren, M. J. L. Quasi-three-dimensional aerodynamic solver for multidisciplinary design optimization of lifting surfaces. *Journal of Aircraft* **51**(2), 547–558 (2014).

[16] Elham, A. and van Tooren, M. J. L. Beyond quasi-analytical methods for preliminary structural sizing and weight estimation of lifting surfaces. In *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 0399, (2015).

[17] Pfeiffer, T., Nagel, B., Böhnke, D., Rizzi, A., and Voskuijl, M. Implementation of a heterogeneous, variable-fidelity framework for flight mechanics analysis in preliminary aircraft design. In *60. Deutscher Luft- und Raumfahrtkongress*, (2011).

[18] Macquart, T., Werter, N., and De Breuker, R. Aeroelastic tailoring of blended composite structures using lamination parameters. In *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 1966, (2016).