CMDOWS: a proposed new standard to store and exchange MDO systems

van Gent, Imco; La Rocca, Gianfranco; Hoogreef, Maurice

DOI
10.1007/s13272-018-0307-2

Publication date
2018

Document Version
Final published version

Published in
CEAS Aeronautical Journal

CEAS Aeronautical Journal (2018) 9:607–627
https://doi.org/10.1007/s13272-018-0307-2

ORIGINAL PAPER

CMDOWS: a proposed new standard to store and exchange MDO systems

Imco van Gent[1] · **Gianfranco La Rocca**[1] · **Maurice F. M. Hoogreef**[1]

Received: 23 November 2017 / Revised: 26 March 2018 / Accepted: 18 April 2018 / Published online: 23 May 2018

Abstract
This paper proposes a new format to store and exchange multidisciplinary design optimization (MDO) systems. Here, the generic term MDO system refers to the set of disciplinary tools, their exchanged data and process connections that, all together, define an MDO computational setup. In the process leading to the formal specification of such a computational system, the set of tools, data and connections evolves, until a complete MDO system formulation (not executable) is reached. The proposed open-source standard, called CMDOWS (Common MDO Workflow Schema), has been developed to support this process. The key aspect of the format is its neutral XML-based data representation, making any stored MDO system exchangeable between the design team members and applications (e.g., tool repositories, visualization packages) developed to support the team in setting up the MDO system. This exchangeability is a key enabler for the creation of a versatile MDO framework. Furthermore, CMDOWS provides the starting point to translate any MDO system formulation into an executable workflow using a workflow platform of choice. To the authors' knowledge, such an exchange format does currently not exist, notwithstanding the enormous potential it would have for the exploitation of large-scale MDO in industry. A case study demonstrating the use of CMDOWS is presented in this paper. It was concluded that the current version of CMDOWS already provides a robust standard to store and exchange MDO systems. The schema will be extended to meet future developments and promote its adoption as a recognized standard in the broader MDO community.

Keywords MDO · CMDOWS · Standardization · Workflow schema · XML · MDO framework

Abbreviations

AGILE	Aircraft 3rd generation MDO for innovative collaboration of heterogeneous teams of experts
BPMN	Business process model and notation
CMDOWS	Common MDO workflow schema
CPACS	Common parametric aircraft configuration schema
DLR	German Aerospace Center
DUT	Delft University of Technology
FMI	Functional mock-up interface
IDEaliSM	Integrated distributed engineering services framework for MDO
KADMOS	Knowledge and graph-based agile design for multidisciplinary optimization system
InFoRMA	Integration, formalization and recommendation of MDO architectures
MDO	Multidisciplinary design optimization
MDF	Multidisciplinary feasible
NLR	Netherlands Aerospace Center
PIDO	Process integration and design optimization
RCE	Remote component environment
SMR	Surrogate model repository
UID	Unique identifier
VISTOMS	Visualization tool for MDO systems
XDSM	Extended design structure matrix
XML	Extensible markup language
XSD	XML schema definition

The research presented in this paper has received funding from the European Union Horizon 2020 Programme (H2020-MG-2014-2015) under Grant agreement no. 636202.

✉ Imco van Gent
i.vangent@tudelft.nl

Gianfranco La Rocca
g.larocca@tudelft.nl

Maurice F. M. Hoogreef
m.f.m.hoogreef@tudelft.nl

[1] Faculty of Aerospace Engineering, TU Delft, Kluyverweg 1, 2629 HS Delft, The Netherlands

1 Introduction

Multidisciplinary Design Optimization (MDO) is a design methodology aimed at capturing and exploiting disciplinary interactions to improve multidisciplinary designs using special mathematical formulations and computational structures. Within the aeronautic community, MDO is considered an extremely high-potential discipline, both for improving the performance of current aircraft designs and supporting the development of future configurations. However, so far, MDO has primarily been demonstrated in literature on academic problems [1–5], while both technical and non-technical barriers [6–10] have limited its adoption by design engineers in industry.

The transformation of MDO from a high-potential discipline into a wide-spread commodity design method in an industrial setting entails a paradigm shift to which it can be hard to adapt, especially when this would affect the modus operandi of large and heterogeneous design teams. Whereas most of the current design methods allow the autonomous analysis of different disciplines (by manually updating values coming from other disciplines), MDO requires the different disciplines to be coupled together in one automated multidisciplinary analysis chain [11]. It is the creation and management of this automated chain, going outside (and often off-sight) the conventional boundaries of discipline competence, that forms a big hurdle in the application of MDO in a collaborative environment. In a survey [12] of their collaborative MDO projects over the past decade, the German Aerospace Center (DLR) found that around 60–80% of the project was generally used to set up the automated chain, which did not include any optimization yet. Several other literature sources [11, 13] confirm that the so-called formulation phase is consuming the most project resources and is the most complex to handle, rather than the actual execution of the numerical optimization. The main stages within the formulation and execution phases of a typical MDO project are summarized in Fig. 1, which will be used throughout this paper to provide context to the addressed research developments.

One of the most interesting outcomes from a survey of recent MDO literature is that, despite the critical hurdle provided by the formulation phase of an MDO system, most of the research seems rather concerned with the execution phase of the MDO system development process. Plenty of literature is also available assessing the benefit of the MDO methodology in different application areas, ranging from aircraft design [14–17] to spacecraft design [18, 19] and the design of wind turbines [20–22].

Many recent publications are also available on new optimization algorithms and in particular on the development of advanced computation infrastructures to exploit distributed computing power, e.g., cloud computing. A limited amount of recent developments relevant to the formulation phase of MDO systems can be found in literature. These include the derivation of new MDO architectures, such as [23, 24], that are not included in the exhaustive overview published by Martins and Lambe [25]. On the other hand, it is peculiar to observe that no new developments can be found in any of the commercial process integration and design optimization (PIDO) platforms (the commodity tools for the execution of MDO workflows), for what concerns their capability to support users in the formulation of MDO problems. As identified by Hoogreef [26], there is no PIDO system that is able to advice or support in the selection of an appropriate MDO architecture for the problem at hand, the problem formulation according to this architecture, nor in the automatic integration of executable MDO workflows according to a given MDO architecture.

A limited amount of literature has been found on specific research to support the formulation of MDO problems and their reconfiguration according to different architectures. Some dedicated languages and grammar have been proposed such as REMS by Alexandrov and Lewis [27, 28] and Ψ by Tosserams [29]. In addition, a few frameworks exist that support the modelling process of MDO architectures, such

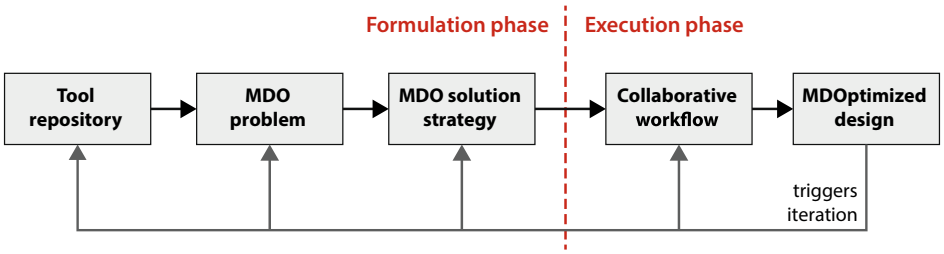


Fig. 1 Different stages of the MDO system in a typical MDO project. In the formulation phase (left) the system is specified by going from a repository of tools to a strategy to solve the MDO problem. The execution phase (right) contains the executable instance of the formulated solution strategy and its result. Usually, the found optimal design triggers an iteration which requires an adjustment of the MDO system at one of the earlier steps to further improve the design

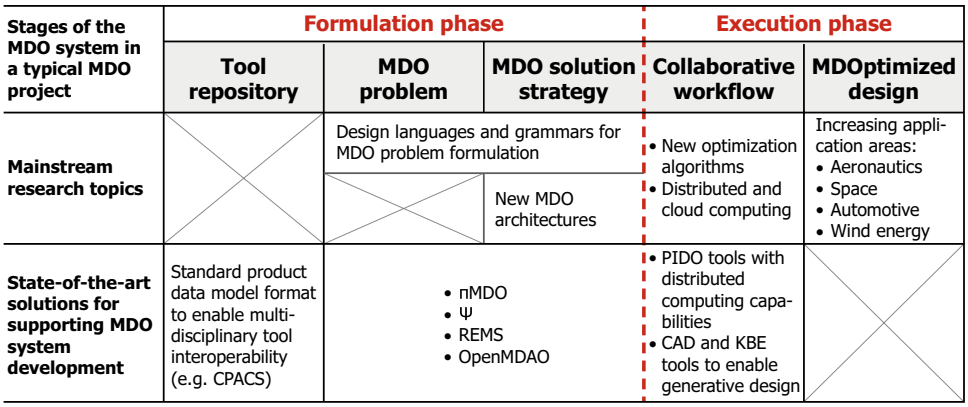


Fig. 2 Mapping of the mainstream research topics and state-of-the-art solutions in the field of MDO on the MDO system development process in Fig. 1

as πMDO by Marriage [30] and the more recent and popular open-source suite OpenMDAO[1] [31].

The outcome of this investigation on recent research developments and state-of-the-art tools to support the development of MDO systems is summarized in Fig. 2. It can be concluded that nothing is available to support the automatic execution (but not even some form of smooth integration) of all the stages specified in Fig. 1. As a consequence, a significant amount of manual, error-prone, time-consuming and repetitive work is left to MDO specialists and system integrators to set up a working MDO system, with the consequences discussed at the beginning of this section.

1.1 New initiatives to support the development of MDO systems

Delft University of Technology (DUT) was involved in two international research projects that aim at supporting the development of MDO systems by strongly reducing their setup time. One is the EU-funded project AGILE[2] (Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts), where one of the main goals was to reduce by 40% the formulation phase of large collaborative MDO systems for aircraft design. Similar objectives were pursued within the ITEA project IDEaliSM[3] (Integrated and Distributed Engineering Services framework for MDO), where partners, from both the aeronautic and automotive supply chain, developed tools and methods to enable the integration of distributed and collaborative MDO frameworks. In both projects, MDO support frameworks were developed, which included various and different applications, that can be grouped according to the following five categories:

Tool repositories A tool repository is a database that contains the definitions of a collection of design and analysis tools which can be made available to the design team to perform MDO. The repository does not necessarily contain the tools themselves, as the sharing of the tool might be prohibited by intellectual property restrictions. In that case, a repository contains the specification of the (interlinked) inputs and outputs of the tools and the way in which each tool can be (remotely) executed. In AGILE, a very convenient approach to assemble large tool repositories, also with tools with many inputs and/or outputs, has been devised based on the use of a central data schema to which each tool input and output is interlinked. Being the focus of AGILE on aircraft design, the DLR standard data model for aircraft CPACS (Common Parametric Aircraft Configuration Schema) [32] has been adopted as central data schema. Using this approach, multiple tool repositories (e.g., a repository containing disciplinary analysis tools and a second repository containing surrogate models) can be combined in a single workflow schema file (so-called repository connectivity graph; details later in this paper), as long as the interlinking is valid. Remote execution of the tools is supported by the collaborative architecture developed in AGILE [33].

MDO system formulation applications The platforms InFoRMA (Integration, Formalization and Recommendation of MDO Architectures) [26] and KADMOS (Knowledge- and graph-based Agile Design for Multidisciplinary Optimization System) [34], both developed by DUT within the IDEaliSM and AGILE projects, respectively, provide two innovative approaches to address the aforementioned challenges of automatic MDO architecture (re)configuration and integration into PIDO software, based on the specification of the general MDO problem definition. Although the implementation and backbone technologies differ drastically for both platforms (KADMOS is Python-based and uses the NetworkX graph package, while InFoRMA is Java-based and uses semantic web technologies), they both use essentially the same

[1] http://openmdao.org, accessed March 15th 2018.

[2] http://www.agile-project.eu, accessed March 15th 2018.

[3] https://itea3.org/project/idealism.html, accessed March 15th 2018.

construct to represent an MDO system throughout the formulation phase: graphs [13, 35]. Earlier work [26, 36] in IDEaliSM has shown that the use of an MDO system formulation platform, such as InFoRMA, can result in a significant setup time reduction, even larger than 90%.

Visualization packages The visualization of large MDO systems can be challenging, but is crucial to share and discuss the design developments within the heterogeneous team of experts. A visualization package to inspect and communicate the workflow schema files produced by the MDO system formulation applications described above can also contribute to decreasing the setup time and definitely increases the trust of the design team in the large, complex automated analysis chain that is being built. Many different forms of visualizations suitable to MDO developments exist, as discussed by Aigner et al. [37], including the well-known XDSM (Extended Design Structure Matrix) [38]. Although both KADMOS and InFoRMA provide some static MDO visualization capabilities, in AGILE a dedicated web-based visualization tool, called VISTOMS (VISualization TOol for MDO Systems) has been developed, which can provide various dynamic and scalable visualizations [39] via web pages.

Collaborative workflows These workflows are the executable instances of the MDO solution strategy produced during the MDO system formulation phase (Fig. 1). The term collaborative is used to express the fact these workflows combine different disciplinary subworkflows from the tool repository, which are owned by different disciplinary experts (or teams), into one optimization workflow. The combination of such subworkflows can be very challenging, especially when the disciplinary teams are distributed either geographically, digitally (i.e., subworkflows running on different server domains), or both. Two main PIDO tools are used in AGILE and IDEaliSM to assemble the executable MDO workflows: Optimus by Noesis Solutions[4] and the Remote Component Environment (RCE)[5] by DLR [40]. Both platforms have been specifically augmented in these projects to enable the automatic generation of executable workflows based on the MDO solution strategy graph produced by KADMOS and InFoRMA [26, 41]. Within AGILE the interoperability of cross-organizational tools (with access restrictions due to intellectual property) in the same executable workflow is supported by the Netherlands Aerospace Center (NLR) tool BRICS [42].

Schema operations library This category contains the collection of useful methods to inspect, check, or analyze the workflow schema file produced by the MDO system formulation applications. It contains functions to check files for their validity (e.g., with respect to the schema definition), to determine key values (e.g., number of tools, number of parameters), and to edit instances of the schema file (e.g., by removing or adding tools and parameters). In AGILE most of these libraries are provided by KADMOS and they are equivalent to some of the functions in the TiXI[6] and TiGL[7] libraries used to inspect and adjust CPACS files.

All the new developments discussed so far are summarized in Fig. 3, where, similarly to Fig. 2, each one of the presented MDO support framework applications is positioned with respect to the five main stages of the MDO system development process presented in Fig. 1.

1.2 Origins of CMDOWS: why a standard to store and exchange MDO systems?

In the early phases of the AGILE and IDEaliSM project, the various applications used to support the development of an MDO system (from the five categories discussed in the previous section), were coupled as illustrated in Fig. 4a. Indeed most of the applications had to communicate directly with each other, with obvious problems of flexibility and maintainability of the overall MDO support framework, due to the many ad-hoc interfaces. Besides, it was realized that the benefits (in terms of overall MDO system setup time reduction) of the vendor-neutral graph-based representations of the MDO workflows produced by KADMOS and InFoRMA were limited without the possibility to automatically generate the executable workflows using a PIDO tool of choice. Also, the investment in the development of the visualization packages would not have been worth it for one specific MDO system formulation tool. On the other hand, the ad-hoc development of visualization capabilities both inside InFoRMA and KADMOS was also not feasible within the project time frame.

Eventually, also on the basis of the evident benefit provided by the CPACS-based central data repository, the authors developed the conviction that a dedicated standard format to define, store and exchange MDO systems would provide a key enabler for the automation of the entire formulation and execution phase of any large collaborative MDO system.

The need of such an exchange format was first advocated within IDEaliSM to facilitate the translation of the MDO system formalization generated by InFoRMA into

[4] https://www.noesissolutions.com/our-products/optimus, accessed March 15th 2018.

[5] http://rcenvironment.de, accessed March 15th 2018.

[6] https://github.com/DLR-SC/tixi, accessed March 15th 2018.

[7] https://github.com/DLR-SC/tigl, accessed March 15th 2018.

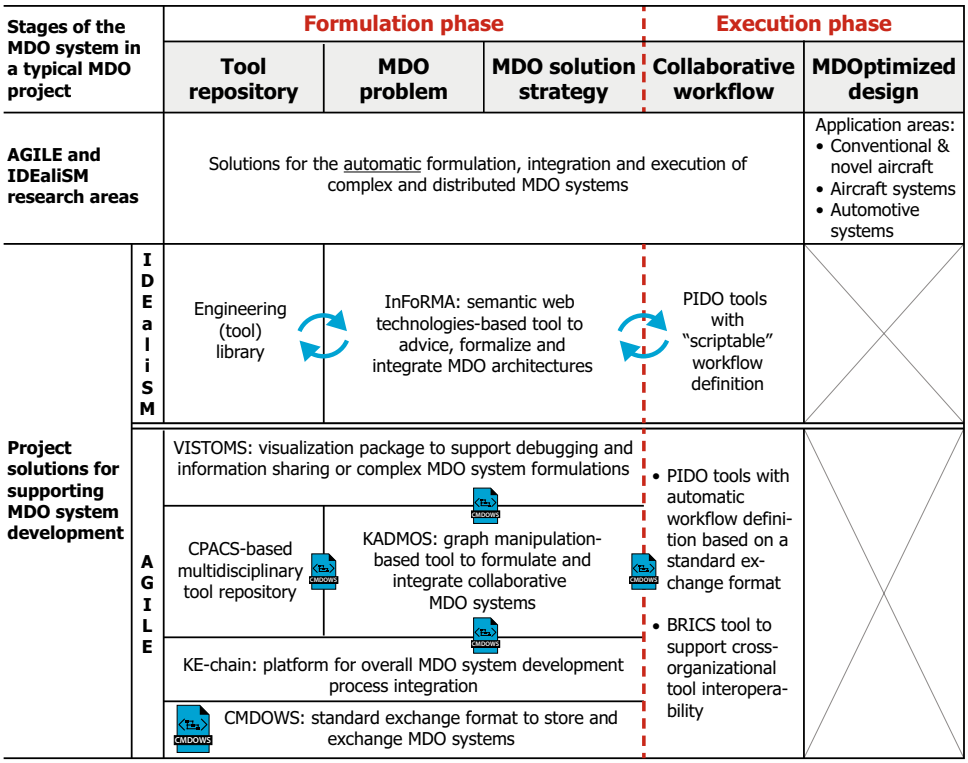
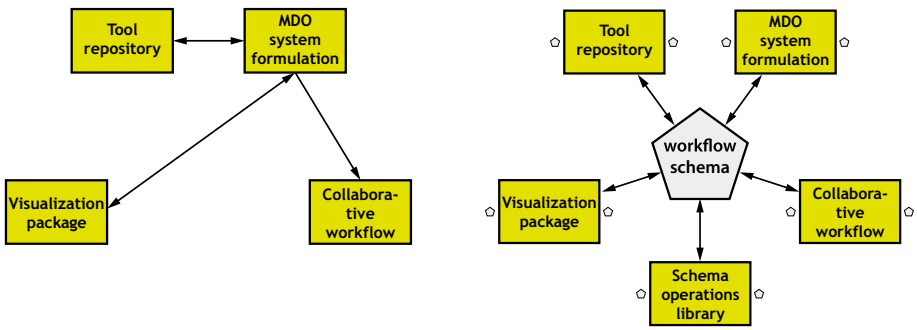


Fig. 3 Mapping of the research activities within the projects AGILE and IDEaliSM on the MDO system development process in Fig. 1



(a) Direct coupling approach **(b)** Central workflow schema approach

Fig. 4 Links between the MDO system formulation platform and other MDO framework categories with two different approaches

executable workflows in Optimus. To this purpose a prototype neutral format was defined by Hoogreef [26]. However, it is in AGILE that the full development of a neutral format for storage and exchange of MDO systems has taken place. The result is the CMDOWS format, which stands for Common MDO Workflow Schema and is the main subject of this paper (see also Fig. 3, bottom).

The need for standardization in engineering design is, of course, not something new. Definitions of standardized formats can be found in earlier work and are usually motivated by similar requirements as the CMDOWS format. For example, Gondhalekar et al. [43] proposed a neutral format in the context of the "behavioural digital aircraft" project CRESCENDO [44] to exchange computational workflows and demonstrated that through such a format the same workflow can be built in two different workflow platforms. Similarly, another format that is gaining momentum is the Functional Mock-up Interface (FMI) [45]. FMI is a platform-independent standard that is aimed at supporting both model exchange and co-simulation of dynamic models. Both these examples are concerned with the sharing of tools within the collaborative workflow, but do not consider other types of applications used to support the development of MDO systems, as was discussed in the previous section. Other formats related to CMDOWS tackle the neutral description and visualization of processes, as is done with BPMN (Business Process Model and Notation) [46] that is maintained by the Object Management Group. A specific visualization standard that captures both process and data flow is provided by the XDSM by Lambe and Martins [38].

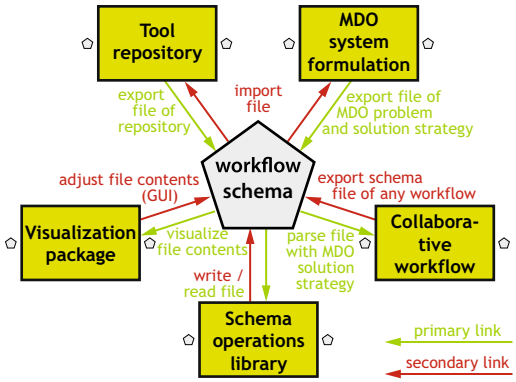


Fig. 5 Primary and secondary links between the workflow schema and the MDO framework application categories

The CMDOWS format proposed in this paper follows the philosophy of the above-mentioned standards, but in the specific context of collaborative MDO projects and with the aim to link together a broader range of applications.

Thanks to the definition of CMDOWS, the application links within the MDO support framework being developed in AGILE have changed from what is shown in Fig. 4a to the new centralized structure shown in Fig. 4b. The graph-based files generated by KADMOS can be stored first as CMDOWS files and then translated into executable workflows by any PIDO tool able to interpret such standard. With the same level of flexibility, the interconnected tool repositories introduced above can be translated into CMDOWS files, which can then be adjusted and enriched by KADMOS, to produce MDO problem graphs (storable again using the CMDOWS standard) and finally MDO solution strategy graphs based on given architectures (again storable using CMDOWS). Eventually, the visualization package, rather than accessing the different internal data structures of KADMOS and InFoRMA, can read and visualize their produced CMDOWS files and help inspecting and monitoring the state of the MDO system during the three stages of the formulation phase.

The five support application categories depicted in Fig. 4b can have bidirectional links to the workflow schema, however, for all of them a primary and a secondary link direction can be identified,[8] as illustrated in Fig. 5. For example, the visualization package has the primary link of being able to open any workflow schema file and depict the visualizations. A secondary link would be in place, if the visualization package would offer users also the possibility to manually edit the visualized CMDOWS file. Generally, the primary link is the one that is most directly useful and, most of the time, also easiest to develop for the category at hand.

1.3 Structure of this paper

This paper is structured as follows: the proposed schema is described in full detail starting from the list of functional requirements in Sect. 2. CMDOWS files generated for a classic MDO benchmark problem are used as examples to illustrate the main branches of the schema. The capability of the proposed schema to support the full development of a complex collaborative MDO system is demonstrated in the study case presented in Sect. 3. This case study concerns the aerostructural optimization of an aircraft wing using the AGILE MDO framework. It includes all the activities from tool repository definition to the generation, based on CMDOWS, of executable workflows for two different PIDO tools. Finally, the main conclusions and an outlook on future developments are given in Sect. 4.

2 CMDOWS

In this section the full CMDOWS definition is described starting from the functional requirements.

2.1 CMDOWS functional requirements

The proposed schema is based on the following nine main functional requirements:

(I) *Machine-interpretable* The format in which the MDO system is stored should be machine-interpretable up to the finest level of detail.

(II) *Human-readable* The schema should allow any designer to inspect at least the top-level correctness of the content, while users and developers with a background in design engineering or computer science should be able to find and understand all the fine details. This human-readability aspect is important to enable the use of the schema by a wider community and to make connecting new MDO framework applications easy.

(III) *Neutral* The schema should not contain elements that are specific to any project, MDO framework application, or developed product. However, the schema should accommodate the storage of any such additional information at specific locations to meet practical issues of certain projects, applications, or products, thereby allowing project-specific additions to the schema file at the dedicated file locations.

[8] N.B. This ordering of the links can be considered subjective and here the ordering is done based on the perspective of the MDO system integrator.

(IV) *Validation* File instances that are based on CMDOWS, should be easily validated against the schema definition.

(V) *Adaptable* From one version release to another, the schema should always be flexible enough to provide room for extensions and enrichment, while at the same time its basic structure should not change too drastically to keep any existing framework application links easily (with a small developing effort) compliant with the release of each new version.

(VI) *Balance of redundant information* Data representation in the schema should aim at minimum redundancy; however, in special cases, this redundancy guideline can be violated for convenience (i.e., to facilitate the link with certain applications that lack the capability to automatically derive the required input based of the information stored in the format). Such redundancies bring the risk of generating inconsistencies in file instances, and therefore, a balance should be found between information that can be implicitly and explicitly stored in the file.

(VII) *Support all MDO system stages* The schema should support storage of the MDO system during the three different stages of the formulation phase, as indicated in Fig. 1.

(VIII) *Support all MDO framework categories* The schema should accommodate all information that is required to enable the links with the five different MDO framework application categories, as depicted in Fig. 5.

(IX) *Support tool heterogeneity* A broad range of analysis tools from the tool repository, including their execution methods, should be stored in the schema, such as simple mathematical expressions, remotely executed 'black boxes', and surrogate models.

With the requirements listed above in mind the CMDOWS version 0.7 has been completed and tested for a realistic aircraft MDO case.

2.2 CMDOWS definition

The eXtensible Markup Language (XML)[9] has been selected as the syntax to store the schema definition. This definition is stored using the XML Schema Definition (XSD)[10] format and this XSD definition of CMDOWS can be used to validate any CMDOWS XML instance, thereby meeting requirement IV (req-IV) on validation in Sect. 2.1. XSD also meets req-I and req-II, as it is both human-readable and supports machine-interpretability. In addition, the XML format is independent of the programming language used. Many programming languages actually include advanced modules to work with the XML format, thereby supporting the human-readability requirement in the sense that many users can easily familiarize themselves with CMDOWS within their preferred programming environment. Another argument for the use of XML is the recent adoption of the XML-based CPACS in the MDO community, meaning that many people are already familiar with the use of XML as a sharable storage format.

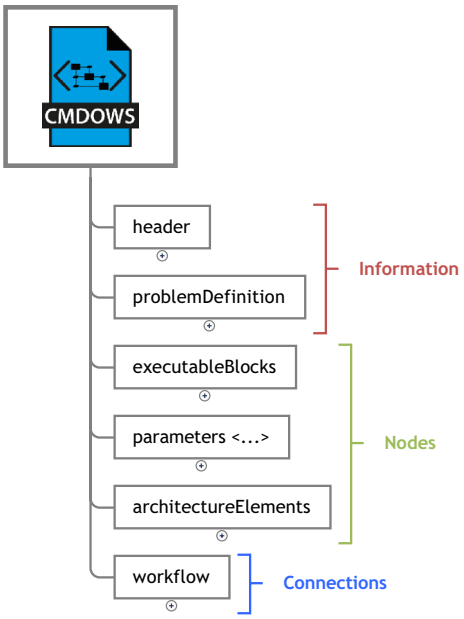


Fig. 6 Top-level elements of CMDOWS and the three main element categories

The CMDOWS definition (see Fig. 6) is structured in six top-level elements, grouped in three basic categories:

- Information
- Nodes
- Connections

This categorization is based on the assumption that any MDO system can be modeled as a graph, as discussed in Sect. 1.1. Graph objects consists of nodes and their connections (also referred to as edges). In the elements of the information category generic information about the graphs is stored, such as its creator, version, and the MDO problem definition used in the second stage of the MDO development process. Node- and connection-specific information is stored as metadata within the respective node and connection

[9] https://www.w3schools.com/xml/default.asp, accessed March 15th 2018.

[10] https://www.w3schools.com/xml/schema_intro.asp, accessed March 15th 2018.

categories. Each category will be discussed in a separate section.[11]

An important concept that is used at different locations in the schema is the separation between parameters and executable blocks. Any node element describing a tool repository, MDO problem, or MDO solution strategy (see Fig. 1) will fall under one of these two groups. The parameters group refers to all the elements inside an MDO system that are assigned a certain value. Parameters are the inputs and/or outputs of the executable blocks, such as the actual optimization parameters (whose values remain constant during optimization) and the design variables (including both actual design variables and the copy or surrogate variables introduced by different MDO strategies). The executable blocks are defined as elements that take certain inputs, perform an operation, and finally produce certain outputs. Since the distinction between parameters and executable blocks is a key aspect, the element names `parameters` and `executableBlocks` appear at different levels of the schema.

For example, a parameter `x1` is input to an executable block and will be defined in the main `parameters` element of the nodes category. Generic information about the parameter, such as a description, unit, and data type would be stored directly on the element as metadata. When this parameter has to be indicated as a design variable (including bounds and nominal value) for a certain MDO system, then this information is stored inside the `problemFormulation/parameters` element. This way, the elements in the nodes category remain independent and valid for any MDO system, while the `problemFormulation` element contains information that is specific to the MDO problem stage from Fig. 1. The two elements are linked together through unique identifiers (UIDs) as will be shown in more detail in the next sections.

2.2.1 Top-level elements

The six top-level elements from Fig. 6 are discussed in more detail in this section.

Elements in the information category The information elements of CMDOWS are `header` and `problemDefinition`. Lower level elements of the information elements are shown in Fig. 7. The `header` element contains metadata relative to the CMDOWS file itself, such as the creator, a description, the schema version used, etc.

The definition of the MDO problem to be solved can be stored in the `problemDefinition` element. This element can get a UID assigned as an attribute (indicated with the @ sign in Fig. 7) so that it can be referred to in other parts of the schema. The other two main elements of the problem definition are `problemRoles` and `problemFormulation`. In the `problemRoles` branch all the special parameters of the MDO system get their roles assigned, such as design, objective, constraint, or state variable, including certain parameter settings for the problem at hand (i.e., upper and lower bounds, constraint types). All executable blocks also get a problem role, based on their connections with other blocks and their position with respect to the design variables, see `preDesvarsBlocks` and `postDesvarsBlocks` in Fig. 7. The second branch of the problem definition is the `problemFormulation`, where the specification of the MDO architecture that should be imposed on the MDO problem and the logical order of the executable blocks are stored. This logical order is required to determine (among others) the feedback between different blocks and can also be used to automatically determine problem roles of the executable blocks.

Elements in the nodes category The node elements all represent either parameters or executable blocks and are separated into three subelements: `executableBlocks`, `parameters`, and `architectureElements`. Their subelements are depicted in Fig. 8. The `executableBlocks` element contains the function blocks that are stored in the tool repository. Two main types of executable blocks can be stored inside this element: *mathematical functions* and *design competences*. The mathematical functions are simple executable blocks that evaluate analytic expressions to determine the value of the outputs. These expressions can be stored directly in the `mathematicalFunction` element, thereby storing the full definition of that executable block. Hence, the actual operation performed by the block is stored for mathematical functions. Contrary to this, design competences represent more complex executable blocks where the operation performed by the block is unknown (or at least cannot be stored as simple mathematical expressions). The `designCompetence` element, therefore, stores a block that performs an unknown operation (it acts as a so-called 'black box'). Instead of storing the operation itself, the schema of the `designCompetence` element can accommodate a range of specifications for executing the tool. For example, a design competence can be an integrated analysis tool on the local system, a remotely called execution using a special server integration, a surrogate model, or any other form of computational module present in a collaborative workflow. Additional information about the competences, such as ownership and the outcome of eventual verification steps, can be stored in the `metadata` element, see Fig. 8.

[11] N.B. For brevity, the description in this paper is limited to the top-level elements of the schema up to level 4 (when the root element is considered to be at level 0), but the full schema, which has elements up to level 7, can be inspected at the open-source repository, see: http://cmdows-repo.agile-project.eu.

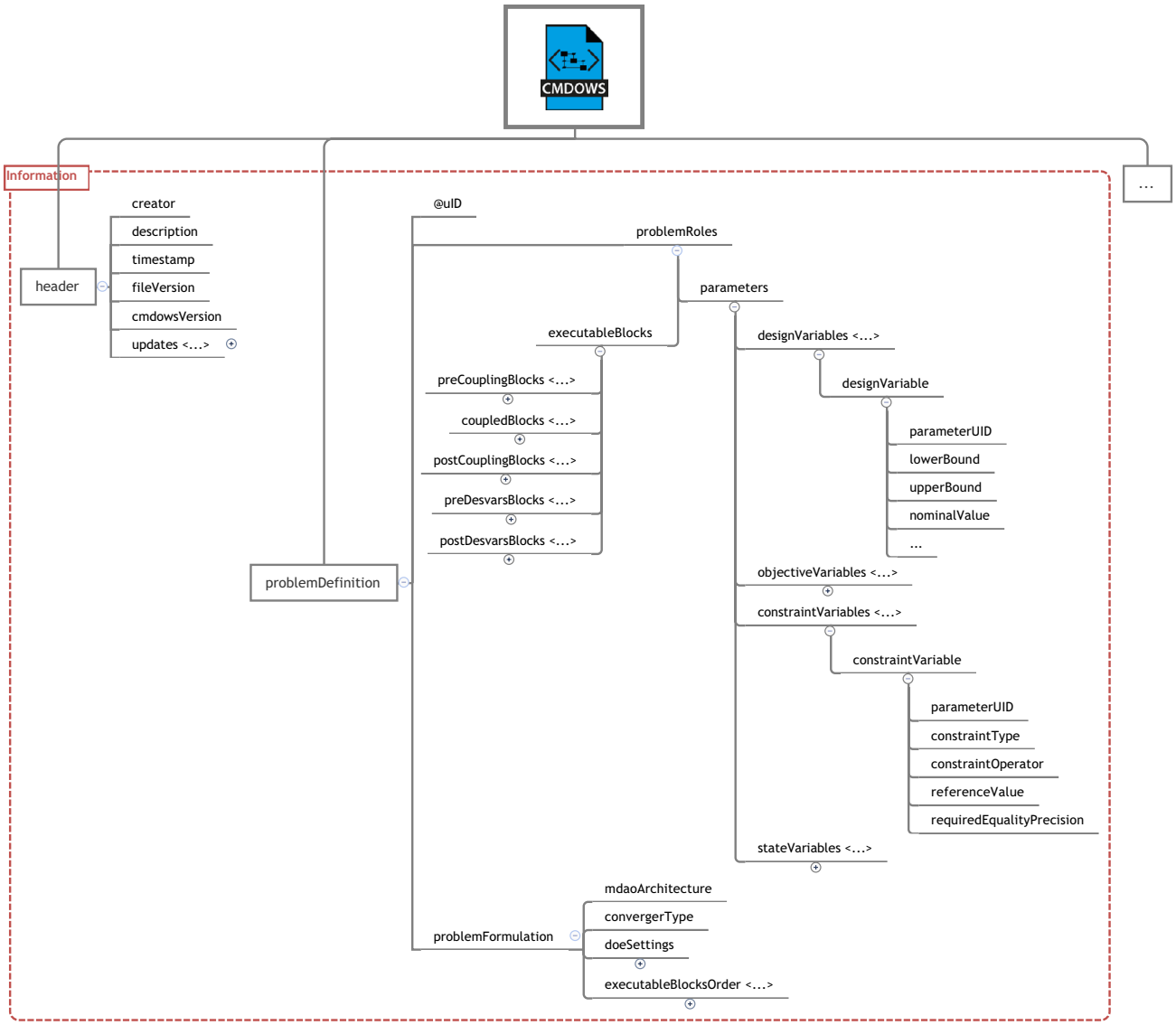


Fig. 7 Elements in the CMDOWS category Information

The second node element is the `parameters` element. This element contains all the inputs and outputs of the executable blocks stored in the main `executableBlocks` element. If the executable blocks are integrated using a central data schema approach (i.e., CPACS), then the `parameters` element will contain all the unique elements that are used from that data schema as separate parameters. Additional information about the parameters can also be stored, such as a label, description, unit, and data type (real, float, list, etc.).

The last node element is `architectureElements`. This element includes both `parameters` and `executableBlocks`, which differ from those stored in the top-level `parameters` and `executableBlocks` element, because they are the additional elements created when an MDO architecture is imposed on a specified MDO problem. For example, when an MDO architecture has to be imposed a new executable block has to be introduced to handle the optimization loop: an optimizer. This new element is stored as an `architectureElements/executableBlock`. New parameters also have to be introduced to connect the optimizer to the rest of the system. Initial guesses for all design variables are required as input of the optimizer and the optimal (final) values of the design variables, objective, and constraints have to be connected as outputs. These parameters do not exist before the imposition of the MDO architecture and are, therefore, instantiated and stored as `architectureElements/parameters`. Similarly, other architecture elements are introduced including convergers, consistency constraint functions and other components

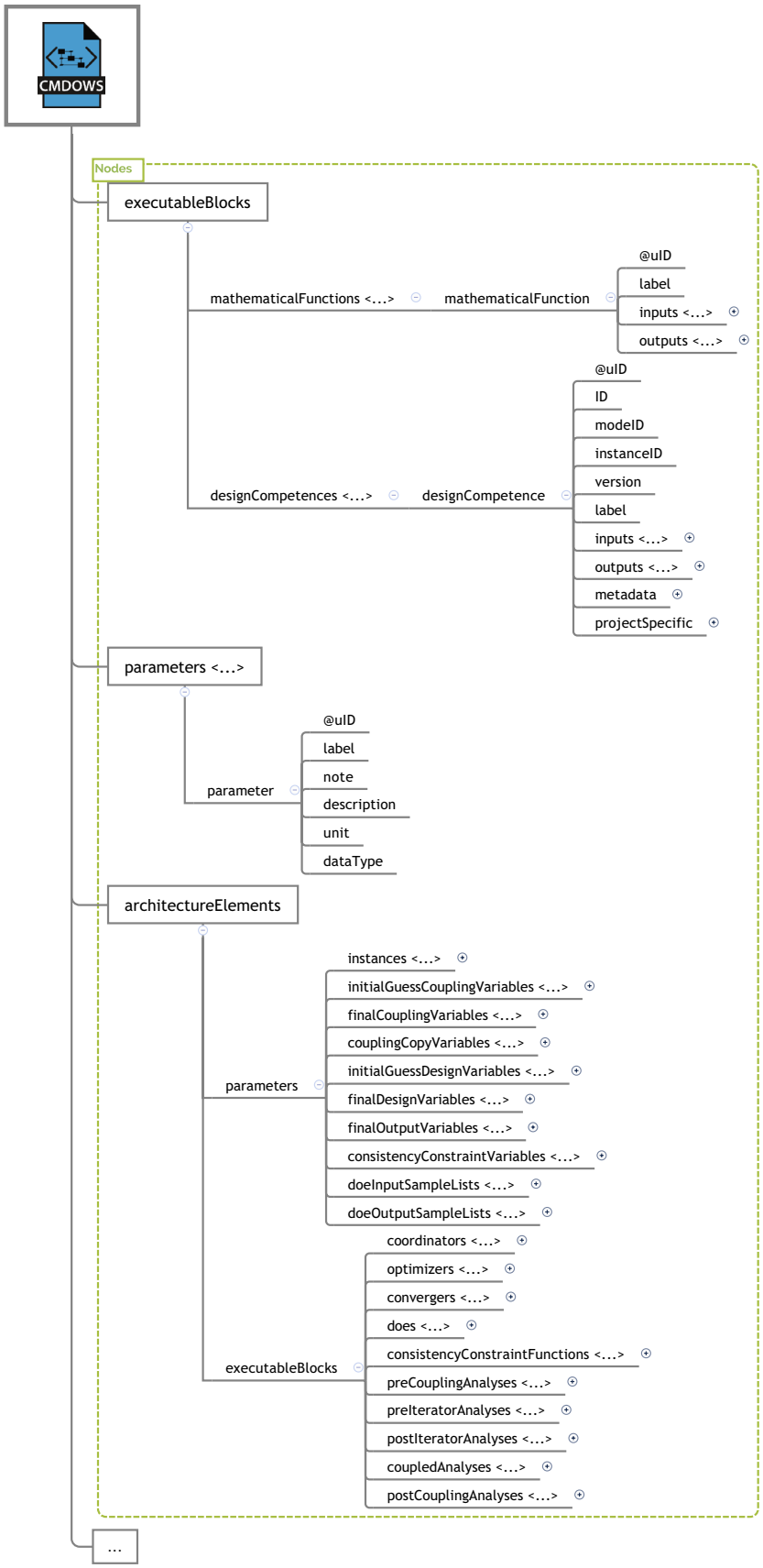


Fig. 8 Elements in the CMDOWS category Nodes

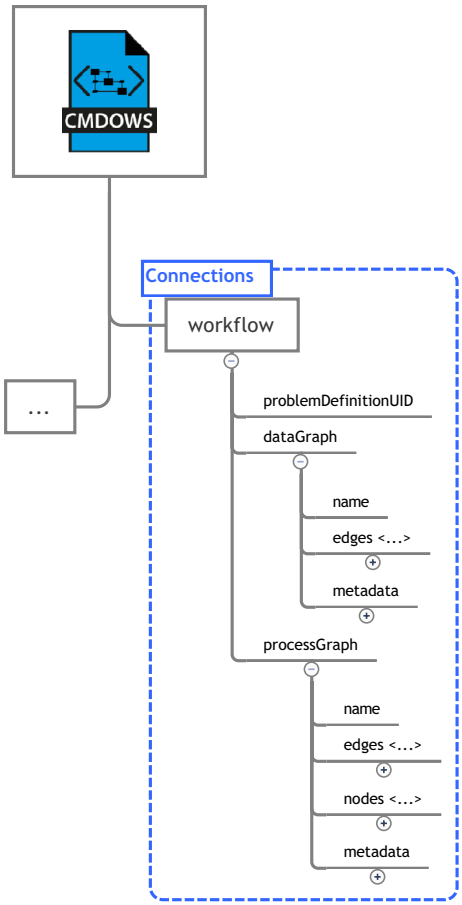


Fig. 9 Elements in the CMDOWS category connections

that are specific to a given MDO architecture. For the complete list of possible architecture elements in the schema, see Fig. 8. Next to newly introduced elements, the original executable blocks are also grouped into different types based on their relation to the coupled system by referring to their UID, see `preCouplingAnalyses` element and below in Fig. 8. This categorization can, for example, be used for assigning colors in the visualization of the CMDOWS file or when parsing the file to create an executable workflow.

Elements in the connections category CMDOWS contains a single element for storing the connections: `workflow`. In this element two different types of graphs can be stored: data graphs and process graphs. The combination of these two graphs constitutes the neutral definition of a workflow that needs to be executed to solve an MDO problem: the MDO solution strategy. The `dataGraph` element contains a data graph storing the connections (or edges) between parameters and executable blocks according to their input/output relations. This data graph can be stored for any stage of the MDO system in Fig. 1, where each stage would be represented by a separate CMDOWS file. The `processGraph` element is only used for the MDO solution strategy to store the process steps for running the different executable blocks. Metadata about the graphs can also be stored, such as the amount of nodes and edges, and the nesting of the process steps for the process graph (Fig. 9).

2.2.2 Illustrative example: storing the Sellar MDO system

The Sellar problem [47], a classical benchmark MDO problem widely used in MDO literature, was selected to demonstrate the use of CMDOWS as a schema to store the three different stages of the MDO system during the formulation phase. As these stages (see Fig. 1) enrich the CMDOWS file step by step, the way in which each stage should be stored is described here to give the reader a clearer understanding of the different elements of the schema. The way this enrichment is performed is out of the scope of this paper, but the interested reader is referred to Van Gent et al. [34], where this process is discussed in detail using the MDO system formulation tool KADMOS.

Stage I: Tool repository The tool repository for the Sellar problem consists of eight different tools. The original Sellar problem actually contains only five tools, but here, three fictitious tools (A, D3, and F2) are added to demonstrate how an MDO problem can be based on a subset of tools from the repository. In the top right of Fig. 10 a design structure matrix of the repository is shown, with the eight tools represented by the blocks on the diagonal. The only elements from the schema needed to store a tool repository are the `designCompetences`, `parameters`, and `workflow/dataGraph`. As shown in Fig. 10, the different executable blocks are integrated differently: for the functions A, F1, F2, G1, and G2 mathematical expression are available, while the disciplinary analyses D1, D2, and D3 are design competences, meaning that the mathematical expressions to be executed are assumed to be unknown (for illustration purposes). For all executable blocks the inputs and outputs of each block are defined by referring to the right elements from the `parameters` list using the relative parameter UID (as is shown for parameters z2 and f in the figure). Finally, the `dataGraph` element contains the full graph, as illustrated in the lower right corner of the figure, by listing the edges between all executable blocks and parameters. The storage of the edge x1 → F1 is illustrated in the figure.

Stage II: MDO problem One additional element is required to store the MDO problem in a CMDOWS file: `problemDefinition`, see Fig. 11. In this element, the problem roles and problem formulation are indicated. As shown in Fig. 11, the parameters z1, z2, and x1 get the special role of design variable. Similarly, f is assigned the role of objective for the optimization. The roles of the executable blocks are also specified and only the tools strictly needed to solve the MDO problem have been selected from the tool

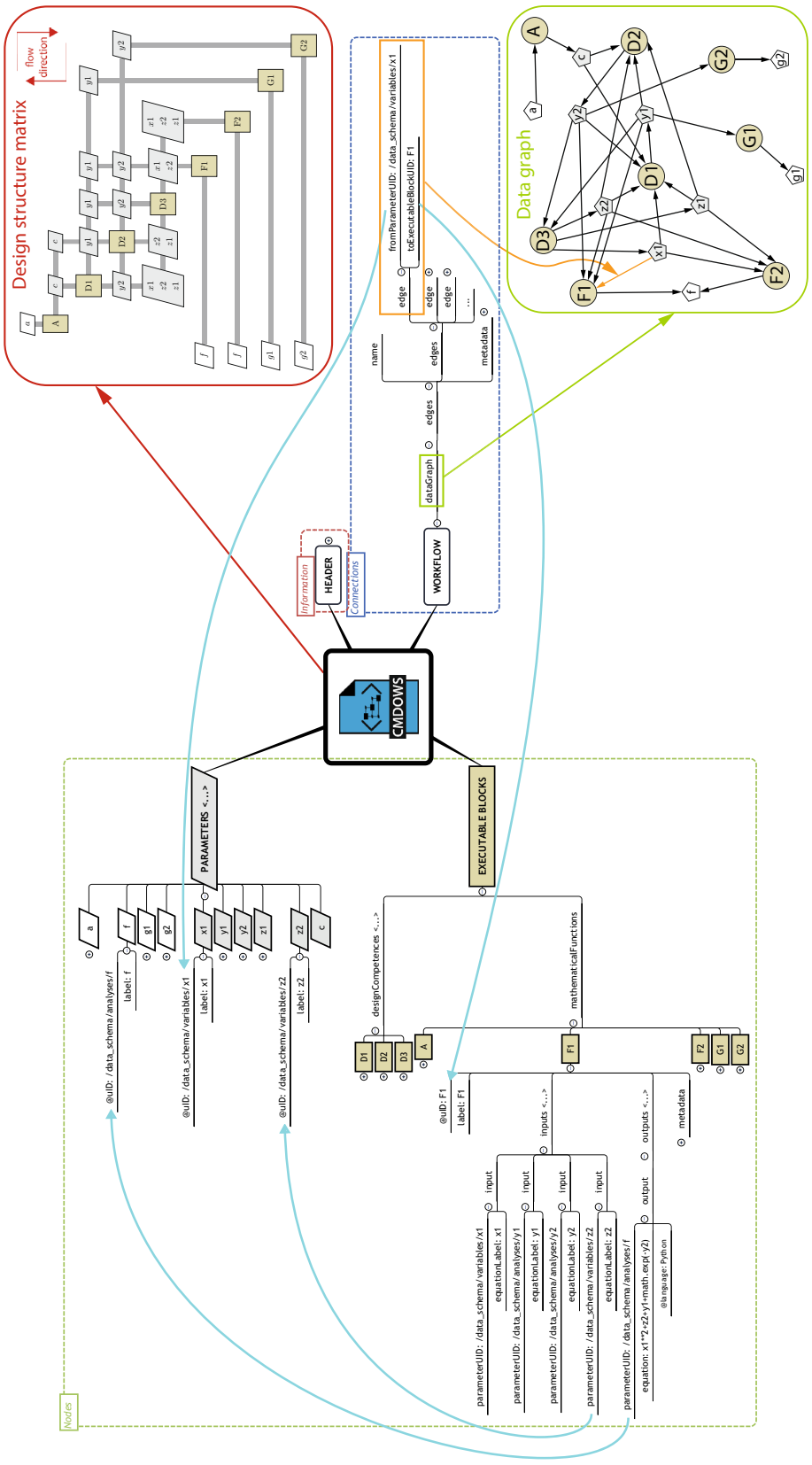


Fig. 10 Illustration of the storage of the Sellar tool repository in a CMDOWS file. On the right side two visualizations of the data stored in the file are shown: the design structure matrix and a directed data graph

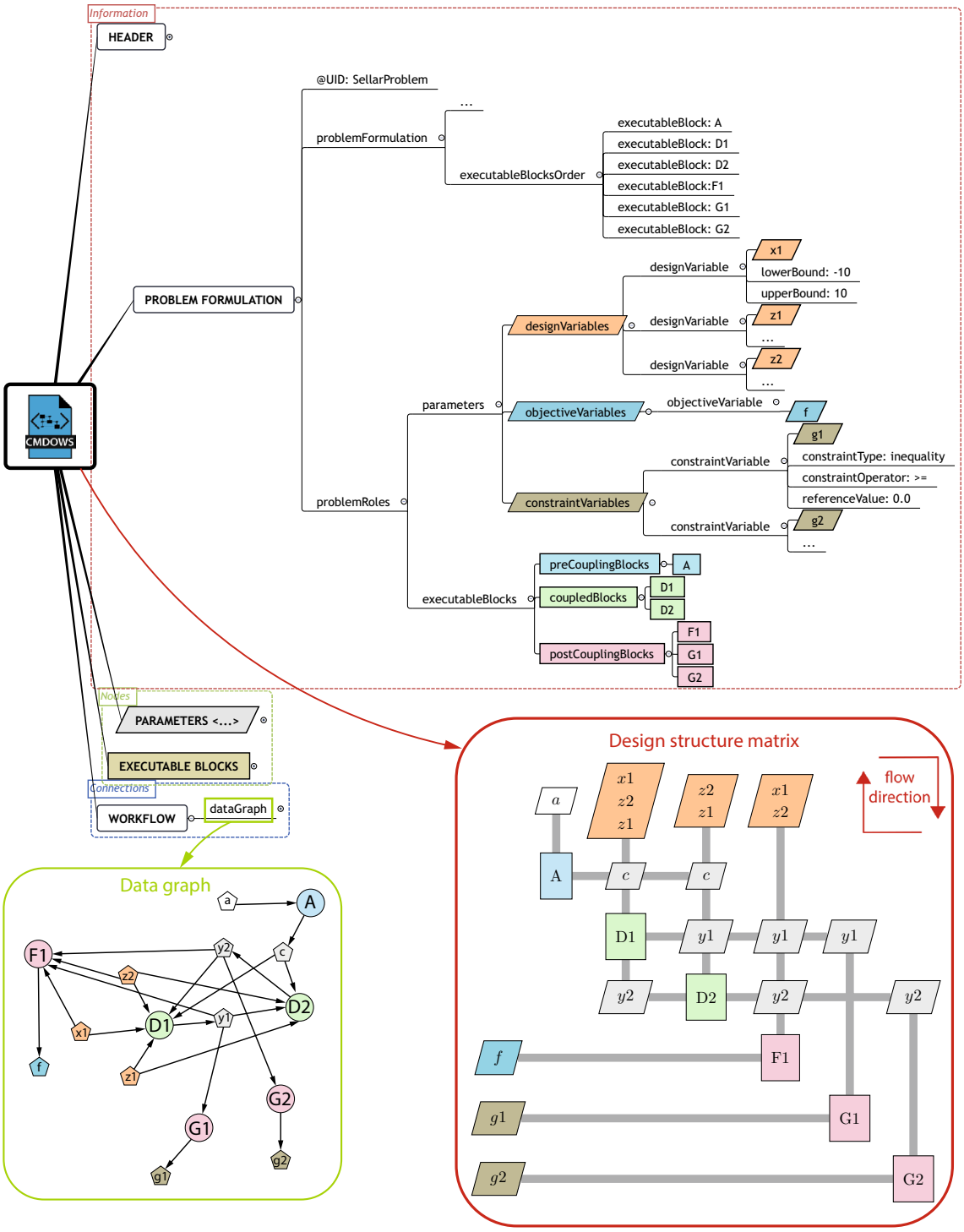


Fig. 11 Illustration of the storage of the Sellar MDO problem in a CMDOWS file

repository, as the tools F2 and D3 are not present in the data graph in Fig. 11 anymore.

Stage III: MDO solution strategy When the problem formulation has been set, in this example to a Multidisciplinary Feasible (MDF) architecture with a Jacobi type converger, the full schema is used to store the MDO solution strategy, as depicted in Fig. 12. This strategy is automatically imposed on the MDO problem using the MDO system formulation platform KADMOS [34]. Two new elements are added to the file with respect to the MDO problem definition: the `architectureElements` and the `workflow/processGraph`. Actually, it is not just that these elements

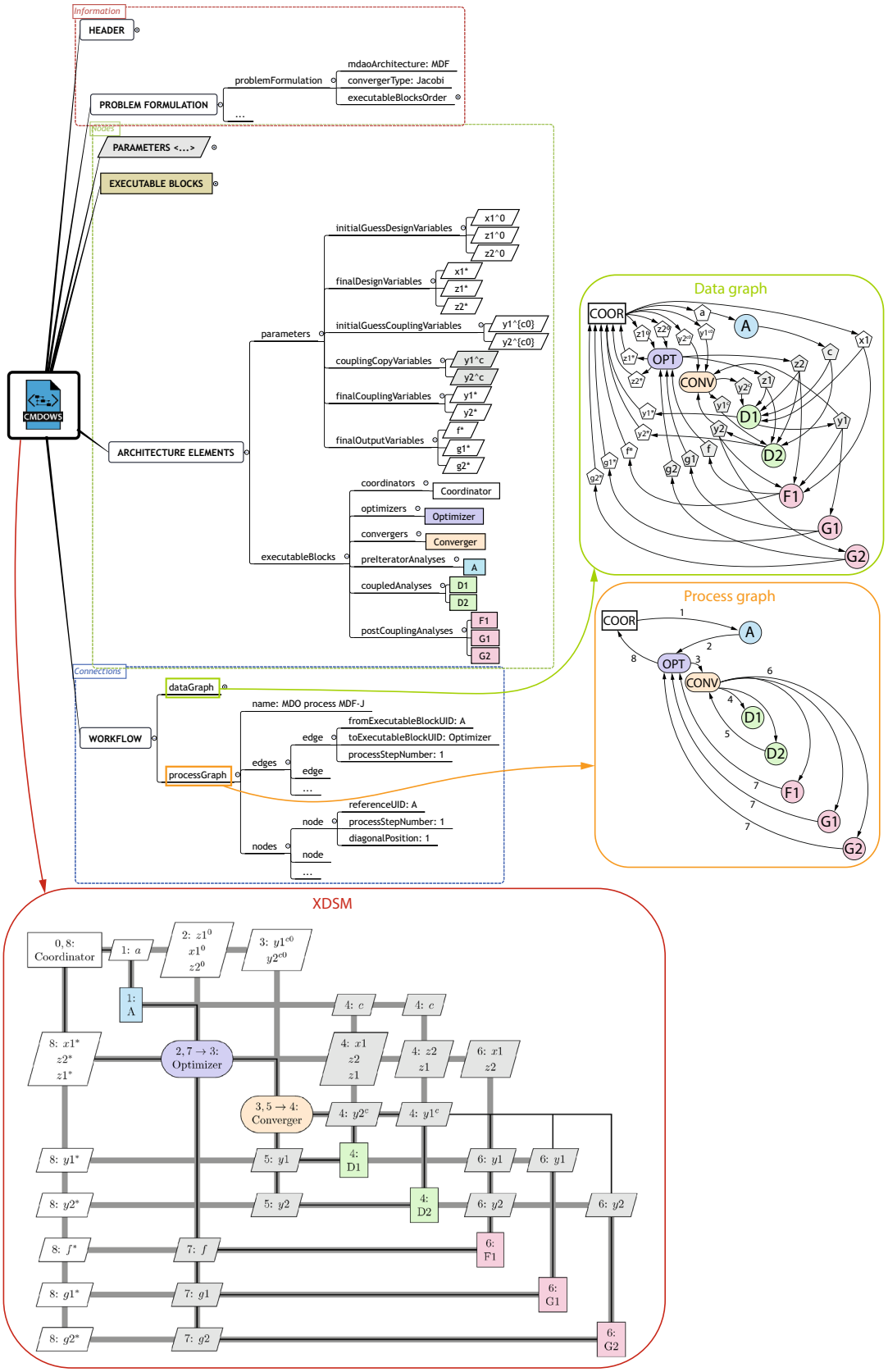


Fig. 12 Illustration of the storage of the Sellar MDO solution strategy according to the MDF-Jacobi architecture in a CMDOWS file

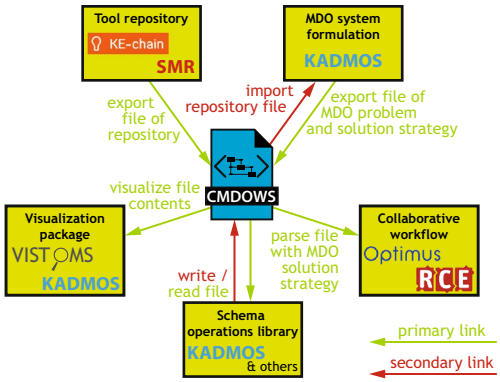


Fig. 13 The established links between CMDOWS and the AGILE MDO framework applications for the wing design case study

are added, but all the elements in the CMDOWS file are updated when the MDO architecture is imposed on the MDO problem. For example, compare the data graphs depicted in Figs. 11 and 12 to see the large amount of adjusted data connections. With the MDF architecture used in this example, the architectural executable blocks `optimizer` and `converger` are added to the file, and a range of architectural parameters are added, such as `initialGuessDesignVariables` and `couplingCopyVariables`.

This concludes a brief illustration of the use of CMDOWS for storing MDO systems at different stages. The use of CMDOWS to store MDO systems has been illustrated here for the small Sellar MDO system. Naturally, the schema was not created for such small cases, but rather to exchange large-scale MDO systems, as discussed in the next section.

3 Case study: AGILE development process for aerostructural wing design

In this case study, CMDOWS files have been created for a realistic wing aerostructural design case. The power of CMDOWS is demonstrated by presenting the different MDO framework applications that were integrated using the central workflow schema approach in the AGILE project.

3.1 Description

Any standardized schema can be put to the test when the exchangeability it is supposed to support can be assessed in a realistic case. This was done here by linking different applications of the AGILE MDO framework through CMDOWS, as conceptually shown in Fig. 5. In AGILE, the five stages shown in Fig. 1 are supported by a set of MDO framework applications, as discussed in Sect. 1.1 (Fig. 3) and more elaborately in earlier work [48]. The AGILE MDO framework is a hybrid framework where different partners provide an application of their specialty with the aim to improve a part of the collaborative MDO design process.

The developments in the AGILE project can be mapped directly on to the conceptual overview in Fig. 5. This is shown in Fig. 13, where at least one application is available for each of the five categories:

- Tool repository:

  KE-chain — The KE-chain platform[12] is used in AGILE to integrate the complete MDO development process, see reference [48]. With respect to CMDOWS, the platform includes a module where a design team can add a collection of disciplinary tools in the browser and can export this online tool repository as a CMDOWS file.

  Surrogate model repository (SMR) — A second type of tool repository is the SMR [42] developed by the NLR. In the SMR a collection of surrogate models is stored which can also be exported as a CMDOWS file.

- MDO system formulation:

  KADMOS — KADMOS is the only MDO system formulation platform that supports CMDOWS at the moment. KADMOS can import CMDOWS files at any stage, transform the MDO system to other stages of the formulation phase, and export the CMDOWS file of an updated MDO system definition.

- Visualization package:

  KADMOS — Basic visualizations of CMDOWS files can be provided by KADMOS. This is restricted to static PDF files and graphs, and is thereby only suitable for small MDO systems or to create a top-level overview.

  VISTOMS — More sophisticated, dynamic visualizations can be created by opening a CMDOWS file with VISTOMS. This tool enables the visualization of MDO systems of any size in multiple dynamic overviews that can be inspected up to the finest details. This package is discussed in

[12] https://www.ke-chain.com, accessed March 15th 2018.

full detail by Aigner et al. [37]. Note that the visualizations of CMDOWS files can also be created online at the open-access CMDOWS interface.[13]

- Collaborative workflow:

RCE	RCE is an open-source development by DLR that is used to create collaborative workflows. RCE's latest version contains an extension to import CMDOWS files and directly create an executable workflow.
Optimus	Optimus is a commercial workflow platform for which an extension has been created to import CMDOWS files. This development is discussed in detail in a paper by Van Gent et al. [41].

- Schema operations library:

KADMOS	At the moment KADMOS is the only platform that can both import and export CMDOWS files. Therefore, it can also perform standardized operations on a CMDOWS file. The CMDOWS module of KADMOS can be used to automatically adjust CMDOWS files or to request information stored in the file, such as the amount of executable blocks, the MDO architecture used, etc.
General XML editors/libraries	Instead of using KADMOS, general XML editors or libraries are also used to inspect and adjust CMDOWS files. An example of such a library is the open-source XML interface library TIXI developed by DLR.

3.2 Results

The aerostructural wing design case used here to demonstrate the integrated platforms has been described in other work [26, 34, 37, 41]. In short, the MDO system in this case consists of a collection of CPACS-compatible aircraft design and analysis tools from DUT. From a tool repository of 29 executable blocks and more than 28,000 parameters, an MDO problem with eight executable blocks involving 281 parameters is composed. This MDO problem can then be solved using different solution strategies, where in this case study the solution strategy demonstrated is the MDF architecture with a Jacobi iteration scheme (as was also used in the Sellar case illustration in Sect. 2.2.2).

The CMDOWS-compatible framework applications shown in Fig. 13 are used for different stages of the MDO system in Fig. 1. The use of the applications with respect to the first four stages of the MDO system from Fig. 1 is summarized in Fig. 14.

The tool repository in the first step is provided through the KE-chain integration. The executable blocks are defined by specifying their CPACS input and output files. KE-chain then interprets these files and creates the list of unique parameters. Additional information on the executable blocks (e.g., owner, fidelity level, etc.) is provided through the browser interface. The CMDOWS file of the tool repository exported by KE-chain can be visualized with the VISTOMS application. The VISTOMS application can actually be used for any stage of the MDO system in the formulation phase, as is shown in Fig. 14.

Next, KADMOS is used to formulate the MDO problem that needs to be solved. The tool repository CMDOWS file is imported by the MDO system formulation platform and KADMOS operations are used to transform the repository data graph into the MDO problem representation (more details on this can be found in [34]). KADMOS then exports the CMDOWS file of the MDO problem and VISTOMS can be used again for detailed inspection of the problem formulation. This visualization is generally used to communicate the setup of the MDO problem with the design team.

KADMOS is used a second time for composing the MDO solution strategy. Based on the MDO problem CMDOWS file containing the `problemFormulation` element, KADMOS can impose the solution strategy and provide the full workflow description. KADMOS adds architecture elements to specify the MDO solution strategy according to the selected architecture and stores in the CMDOWS file both the adjusted data graph and the newly generated process graph. At this last formulation stage the VISTOMS package can be used to check the MDO solution strategy with the whole design team, for example using the XDSM shown in Fig. 14.

The ultimate test of the full schema now comes when the gap with the executable phase has to be bridged. Within AGILE, both Optimus and RCE have been extended to enable automatic generation of executable workflows based on CMDOWS files and both have been able to parse executable workflows for the wing design case, as shown in the bottom of Fig. 14. Any other PIDO tool able to do the same, could directly be used interchangeably in the AGILE MDO framework, without any modification to the other applications. The parsing of Optimus workflows is discussed in [41]. With the availability of the executable workflows the role of the CMDOWS format in the formulation phase is concluded.

[13] http://cmdows.agile-project.eu, accessed March 15th 2018.

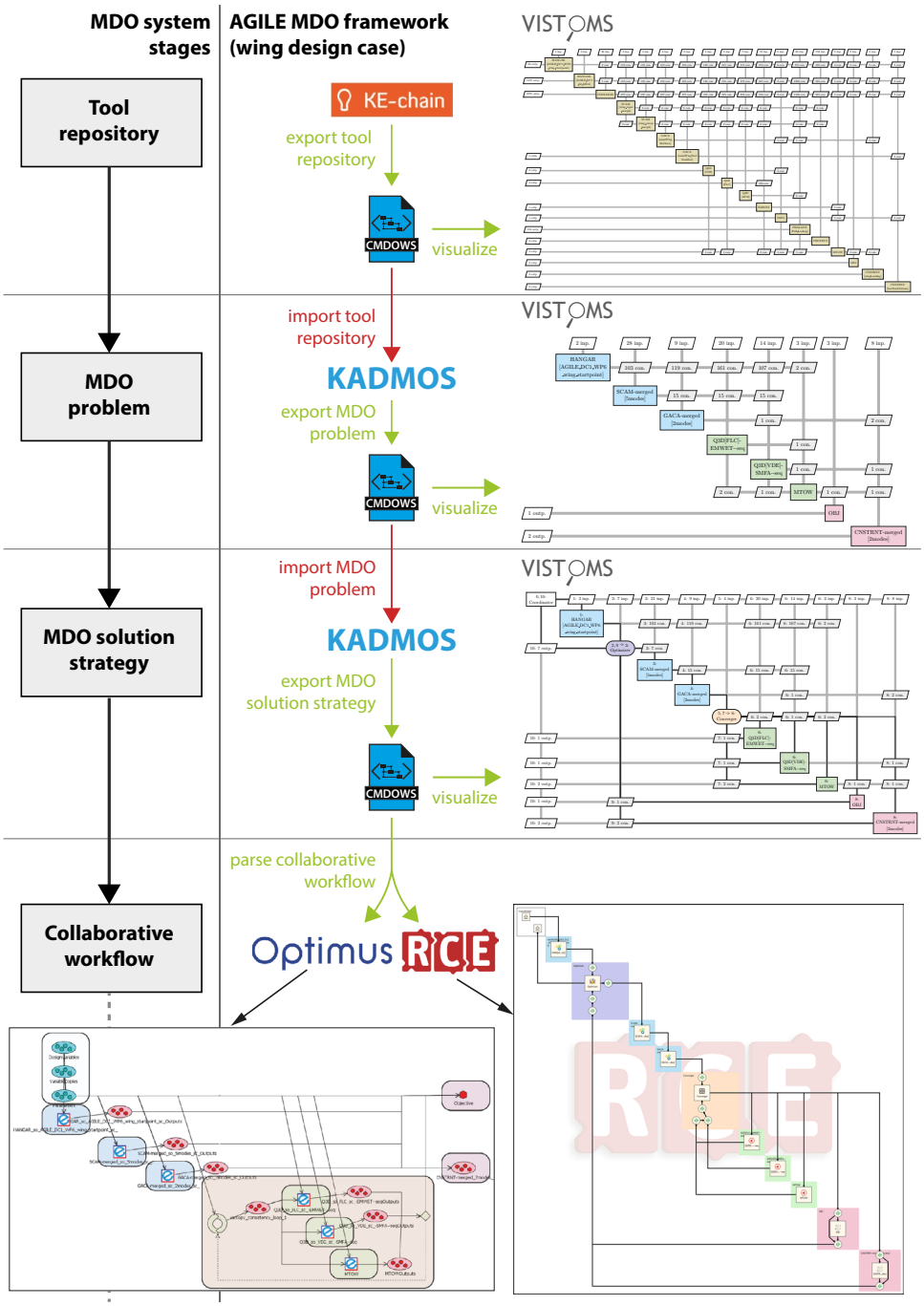


Fig. 14 Visualization of the AGILE MDO framework applications from Fig. 13 for the wing design case, mapped on the first four stages of the MDO system in Fig. 1

3.3 Discussion

Looking back at the functional requirements described in Sect. 2.1, it is important to note that the executable workflows created are truly hybrid workflows. Different types of tools are integrated in one top-level collaborative workflow, thanks to the support of the CMDOWS format.

The first type is used by the three 'blue' tools in the MDO solution strategy in Fig. 14. These are local tools that have been integrated directly in Optimus and RCE. Then the 'green' disciplinary analyses are of another type, as these are subworkflows that need to be executed remotely because of intellectual property restrictions (these tools cannot be distributed to run them locally). Therefore, these disciplinary analysis are integrated using a BRICS [42] component to run the actual tool on another server domain. Finally, the 'red' tools are simple mathematical functions describing the objective and constraint functions. These mathematical functions are parsed as native scripts in the PIDO tool, since this will result in the most efficient execution. The native scripts

run the mathematical expressions directly, without the need to integrate any tool in the collaborative workflow. This collaborative workflow clearly shows that the schema supports the tool heterogeneity requirement IX (req-IX) specified in Sect. 2.1.

Concerning the other requirements in Sect. 2.1, the wing design case study has shown the level of compliance of the current schema. The XML CMDOWS instances support both human-readability (req-II) and machine-interpretability (req-I). This human-readability is also proven by the fact that many of the developers of AGILE framework applications have been able to connect to CMDOWS in a short time. The neutrality of the schema (req-III) has been maintained, even when adding new elements to support the links with the AGILE framework applications. Hence, there are no traces of application-specific elements like KADMOS, KE-Chain, Optimus, etc. Moreover, the core structure of CMDOWS still allows adjustments (req-V), as the schema was extended step by step to link different applications and more adjustments can be made to meet future developments.

A key future improvement that was found concerns the redundancy of the content of the schema (req-VI). Throughout its development, initial CMDOWS versions were always very lean in the information stored in a CMDOWS file. Some of the application links demanded that certain information is stored explicitly in the schema, even though this information can be interpreted from the information already stored. In future developments, the links with applications will be checked for this type of information and per case it will be decided whether to explicitly add the information in the schema.

The wing design case has shown that three MDO system stages (req-VII) are supported and even the bridge to the execution phase can be made successfully. At the moment, not all links are made with the MDO framework applications (req-VIII). The links in Fig. 14 are mostly primary links between CMDOWS and the applications, as explained in Fig. 5. In future work, all applications will be extended and the secondary links will also be developed to enhance the capabilities of the AGILE MDO framework.

The implementation of the CMDOWS format in a heterogeneous MDO framework has enabled the coupling of multiple MDO framework applications that would normally operate independently. The associated time reduction that motivated this development has not been quantified yet, though it is estimated by MDO experts to be significant [10, 13]. For example, the time reduction achieved through InFoRMA for creation of the executable workflows for the same MDO problem, where also the same MDO solution strategy was imposed automatically, were beyond 90% [26] using a prototype version of a standardized format. The quantification of the time reduction impact of CMDOWS within a broader MDO framework will be one of the results of the final year of AGILE, where the framework integration supported by CMDOWS will be put to the test in six MDO studies of unconventional aircraft configurations (e.g., blended-wing body, box-wing). The increased agility of the collaborative MDO framework enabled by CMDOWS is estimated to contribute time reductions in the order of 10–20%. This reduction would be amplified in a true MDO study project, where the assembly of a single MDO strategy is only one of the multiple reconfiguration steps involved. In general, any MDO study starts with simple convergence studies, followed by sensitivity studies aimed at selecting the most convenient set of design variables. Once a given MDO strategy has been implemented, the insights obtained from the execution typically lead to a change or addition of tools, selection of different design variables, objectives and constraints, or the switch to a different architecture. As demonstrated in AGILE [33, 49], CMDOWS is the actual facilitator of these multiple studies providing an essential contribution to the pursued agility.

Although the AGILE design cases strongly benefit from the adoption of CMDOWS, a number of criteria should be considered to evaluate its convenience in a generic MDO study case:

- size of the MDO system under consideration
- heterogeneity and distribution of the team
- heterogeneity of the MDO framework
- maturity of the existing MDO framework with respect to using CMDOWS

When dealing with large (in terms of number of involved tools and coupling parameters) MDO systems, design teams will benefit greatly from adapting CMDOWS to quickly set up a coherent tool repository and use that repository to formulate the MDO problem and solution strategy. Similarly, a heterogeneous and distributed team (many specialists with different backgrounds working at different locations) will benefit from CMDOWS to serve as a "common language" to streamline the definition and use of MDO systems of any size.

The heterogeneity of the MDO framework, as indicated in Fig. 4, has been the main motivation for developing CMDOWS and is also the strongest indication for its effective use. There is a general skepticism in the development and adoption of monolithic, holistic solutions that can cover all the aspects of performing MDO projects collaboratively, e.g., MDO system definition, visualization, problem formulation, reconfiguration and execution. Such solutions, whenever available, become obsolete quickly and are typically inflexible. A simple but comprehensive standard format as CMDOWS, on the other hand, allows the (re-)integration of many different applications, both commercial and in-house developed, thus providing maximum flexibility, scalability and adaptability.

Finally, it is worth noting that not all people involved in a project necessarily need to familiarize themselves with CMDOWS, as this depends on the level of maturity of the MDO framework in using CMDOWS at the beginning of the project. In a mature framework all MDO applications will already use and produce CMDOWS files (see Fig. 5) and most people involved simply use these applications to perform their tasks. Only people involved in developing and maintaining the applications will have to invest time for familiarization. Based on experience in the AGILE project, this familiarization time is limited to 2-3 days with some additional time required to understand the basic concepts of MDO and XML. The AGILE project has proven that CMDOWS interfaces can be easily developed, including parsers for a heterogeneous set of PIDO tools (RCE, Optimus). Once the interfaces are in place, the presence of CMDOWS is transparent to the user, but does not require any direct manipulation nor familiarization with the format itself.

4 Conclusions and future developments

The latest version (0.7) of the MDO system exchange format CMDOWS has been presented in this paper. CMDOWS supports the storage of an MDO system of any size at three different stages of the formulation phase: tool repository, MDO problem, and MDO solution strategy. The main goal of CMDOWS is to provide a format that allows different MDO framework applications to exchange the definition of the MDO system. CMDOWS was demonstrated using an aerostructural wing design problem within the AGILE context. Different AGILE MDO framework applications were linked to CMDOWS and it was shown that all stages of the formulation phase are successfully supported by the schema. Moreover, the final formulation stage results in a CMDOWS file for which a collaborative workflow can be instantiated directly in the workflow softwares Optimus and RCE. This last functionality alone is already sufficient to demonstrate the key role a standard format to store and exchange MDO systems can play in the reduction of the setup time of an MDO system. The enabled check and debugging operations supported by automatically generated visualizations, together with the enabled automated generation of executable workflows alone, can reduce setup time of MDO systems even beyond 90%.

Future work will focus on extending the schema, while still maintaining the nine main requirements stated in Sect. 2.1. As the AGILE MDO framework will grow, so could the schema to support additional or enriched links between CMDOWS and the MDO framework applications. An example of an enriched link would be the future extensions that might be required to support the storage of multilevel optimization architectures, such as BLISS (Bilevel Integrated System Synthesis) [50], since so far the schema has only been tested for monolithic formulations.

In conclusion, the current version of CMDOWS already demonstrated its potential and versatility by demonstrating the connection of five different MDO framework applications and its support for the development of an MDO system from the tool repository to the collaborative workflow stage for a realistic aerostructural wing optimization problem. It is expected that the presented developments will gain enough momentum and lead to a broad adoption of CMDOWS as a standard to store and exchange MDO systems for a large range of MDO framework applications.

Acknowledgements 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 No. 636202. The authors are grateful to the partners of the AGILE consortium for their contribution and feedback.

Open-source references

CMDOWS repository	http://cmdows-repo.agile-project.eu
CMDOWS interface	http://cmdows.agile-project.eu
KADMOS repository	https://bitbucket.org/imcovangent/kadmos
RCE	http://rcenvironment.de

References

1. Kodiyalam, S.: Evaluation of methods for multidisciplinary design optimization (MDO), Phase I, Contractor Report CR-1998-208716. National Aeronautics and Space Administration, Langley Research Center (1998)
2. Kodiyalam, S., Yuan, C.: Evaluation of methods for multidisciplinary design optimization (MDO), Part II, Contractor Report CR-2000-210313. National Aeronautics and Space Administration, Langley Research Center (2000)
3. Brown, N.F., Olds, J.R.: Evaluation of multidisciplinary optimization techniques applied to a reusable launch vehicle. J. Spacecr. Rock. **43**(6), 1289–1300 (2006)
4. Perez, R.E., Liu, H.H.T., Behdinan, K.: Evaluation of multidisciplinary optimization approaches for aircraft conceptual design. AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference (2004)

5. Roth, B., Kroo, I.: Enhanced collaborative optimization: application to an analytic test problem and aircraft design. In: 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference (2008)
6. Belie, R.: Non-technical barriers to multidisciplinary optimisation in the aerospace industry. In: 9th AIAA/ISSMO Symposium of Multidisciplinary Analysis and Optimisation, pp. 4–6 (2002)
7. Giesing, J.P., Barthelemy, J.: A summary of industry MDO applications and needs. AIAA White Paper (1998)
8. Agte, J., De Weck, O., Sobieszczanski-Sobieski, J., Arendsen, P., Morris, A., Spieck, M.: MDO: assessment and direction for advancement—an opinion of one international group. Struct. Multidiscip. Optim. **40**(1–6), 17–33 (2010)
9. Shahpar, S.: Challenges to overcome for routine usage of automatic optimisation in the propulsion industry. Aeronaut. J. **115**(1172), 615 (2011)
10. Simpson, T.W., Martins, J.R.R.A.: Multidisciplinary design optimization for complex engineered systems: report from a national science foundation workshop. J. Mech. Des. **133**(10), 101002 (2011)
11. Flager, F., 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, pp. 625–630 (2007)
12. Ciampa, P.D., Nagel, B.: Towards the 3rd generation MDO collaboration environment. In: 30th Congress of the International Council of the Aeronautical Sciences (2016)
13. Pate, D.J., Gray, J., German, B.J.: A graph theoretic approach to problem formulation for multidisciplinary design analysis and optimization. Struct. Multidiscip. Optim. **49**(5), 743–760 (2014)
14. Meadows, N., Schetz, J., Kapania, R., Bhatia, M., Seber, G.: Multidisciplinary design optimization of medium-range transonic truss-braced wing transport aircraft. J. Aircr. **49**(6), 1844–1856 (2012)
15. Mallik, W., Kapania, R., Schetz, J.: Effect of flutter on the multidisciplinary design optimization of truss-braced-wing aircraft. J. Aircr. **52**(6), 1858–1872 (2015)
16. Iwaniuk, A., Wisniowski, W., Zóltak, J.: Multi-disciplinary optimisation approach for a light turboprop aircraft-engine integration and improvement. Aircr. Eng. Aerosp. Technol. **88**(2), 348–355 (2016)
17. Bach, T., Führer, T., Willberg, C., Dähne, S.: Automated sizing of a composite wing for the usage within a multidisciplinary design process. Aircr. Eng. Aerosp. Technol. **88**(2), 303–310 (2016)
18. Balesdent, M., Bérend, N., Dépincé, P., Chriette, A.: A survey of multidisciplinary design optimization methods in launch vehicle design. Struct. Multidiscip. Optim. **45**(5), 619–642 (2012)
19. Adami, A., Mortazavi, M., Nosratollahi, M.: A new approach in multidisciplinary design optimization of upper-stages using combined framework. Acta Astronaut. **114**, 174–183 (2015)
20. Ashuri, T., Zaaijer, M., Martins, J., van Bussel, G., van Kuik, G.: Multidisciplinary design optimization of offshore wind turbines for minimum levelized cost of energy. Renew. Energy **68**, 893–905 (2014)
21. Ashuri, T., Zaaijer, M., Martins, J., Zhang, J.: Multidisciplinary design optimization of large wind turbines—technical, economic, and design challenges. Energy Convers. Manag. **123**, 56–70 (2016)
22. Ashuri, T., Martins, J., Zaaijer, M., van Kuik, G., van Bussel, G.: Aeroservoelastic design definition of a 20 MW common research wind turbine model. Wind Energy (2016)
23. Jiang, P., Wang, J., Zhou, Q., Zhang, X.: An enhanced analytical target cascading and Kriging model combined approach for multidisciplinary design optimization. Math Probl Eng (2015)
24. Ollar, J., Toropov, V., Jones, R.: Sub-space approximations for MDO problems with disparate disciplinary variable dependence. Struct. Multidiscip. Optim. 1–10 (2016)
25. Martins, J.R.R.A., Lambe, A.B.: Multidisciplinary design optimization: a survey of architectures. AIAA J. **51**(9), 2049–2075 (2013)
26. Hoogreef, M.F.M.: Advise, formalize and integrate MDO architectures—a methodology and implementation. Ph.D. thesis, Delft University of Technology (2017)
27. Alexandrov, N., Lewis, R.: Reconfigurability in MDO Problem Synthesis, Part 1, 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference (2004)
28. Alexandrov, N., Lewis, R.: Reconfigurability in MDO problem synthesis, Part 2, 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference (2004)
29. Tosserams, S., Hofkamp, A., Etman, L., Rooda, J.: A specification language for problem partitioning in decomposition-based design optimization. Struct. Multidiscip. Optim. **42**(5), 707–723 (2010)
30. Marriage, C.: Automatic implementation of multidisciplinary design optimization architectures using π MDO. Master's thesis, University of Toronto, Canada (2008)
31. Gray, J., Moore, K.T., Hearn, T.A., Naylor, B.A.: A standard platform for testing and comparison of MDAO architectures. In: 8th AIAA multidisciplinary design optimization specialist conference (MDO), Honolulu, pp. 1–26 (2012)
32. Nagel, B., Böhnke, D., Gollnick, V., Schmollgruber, P., Rizzi, A., La Rocca, G., Alonso, J.J.: Communication in aircraft design: Can we establish a common language? In: 28th International Congress Of The Aeronautical Sciences, Brisbane (2012)
33. Ciampa, P.D., Baalbergen, E.H., Lombardi, R.: A collaborative architecture supporting AGILE design of complex aeronautics products. In: 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference (2017)
34. van Gent, I., La Rocca, G., 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)
35. Diestel, R.: Graph theory. Graduate Texts in Mathematics, vol. 173 (2010)
36. Raju Kulkarni, A., Hoogreef, M.F.M., La Rocca, G.: Combining semantic web technologies and KBE to solve industrial MDO problems. In: 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference (2017)
37. Aigner, B., van Gent, I., La Rocca, G., Stumpf, E., Veldhuis, L.L.M.: Graph-based algorithms and data-driven documents for formulation and visualization of large MDO systems. CEAS Aeronaut. J. (2018, accepted)
38. Lambe, A.B., Martins, J.R.R.A.: Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. Struct Multidiscip Optim **46**(2), 273–284 (2012)
39. Aigner, B., van Gent, I., La Rocca, G., Stumpf, E., Veldhuis, L.L.M.: Using graph-based algorithms and data-driven documents for formulation and visualization of large MDO systems. In: 6th CEAS Air and Space Conference (2017)
40. Seider, D., Fischer, P.M., Litz, M., Schreiber, A., Gerndt, A.: Open source software framework for applications in aeronautics and space. In: 2012 IEEE Aerospace Conference (2012)
41. van Gent, I., Lombardi, R., La Rocca, G., d'Ippolito, R.: A fully automated chain from MDAO problem formulation to workflow execution. In: EUROGEN 2017 (2017)
42. Baalbergen, E., Kos, J., Louriou, C., Campguilhem, C., Barron, J.: Streamlining cross-organisation product design in aeronautics. Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng. **231**(12), 2192–2202 (2017)

43. Gondhalekar, A.C., Guenov, M.D., Wenzel, H., Balachandran, L.K., Nunez, M.: Neutral description and exchange of design computational workflows. In: 18th International Conference on Engineering Design (2011)
44. Kesseler, E., Guenov, M.D.: Advances in collaborative civil aeronautical multidisciplinary design optimization. In: American Institute of Aeronautics and Astronautics (2010)
45. Blochwitz, T., Otter, M., Arnold, M., Bausch, C., Elmqvist, H., Junghanns, A., Mauß, J., Monteiro, M., Neidhold, T., Neumerkel, D., et al.: The functional mockup interface for tool independent exchange of simulation models. In: 8th International Modelica Conference. No. 063, pp. 105–114 (2011)
46. Grosskopf, A., Decker, G., Weske, M.: The process: business process modeling using BPMN. Meghan Kiffer Press (2009)
47. Sellar, R.S., Batill, S.M., Renaud, J.E.: Response surface based, concurrent subspace optimization for multidisciplinary system design. AIAA Paper **714**, 1996 (1996)
48. van Gent, I., Ciampa, P.D., Aigner, B., Jepsen, J., La Rocca, G., Schut, E.J.: Knowledge architecture supporting collaborative MDO in the AGILE paradigm. In: 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference (2017)
49. Lefebvre, T., Bartoli, N., Dubreuil, S., Panzeri, M., Lombardi, R., Della Vecchia, P., Nicolosi, F., Ciampa, P.D., Anisimov, K., Savelyev, A.: Methodological enhancements in MDO process investigated in the AGILE European project. In: 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference (2017)
50. Sobieszczanski-Sobieski, J., Altus, T.D., Phillips, M., Sandusky, R.: Bilevel integrated system synthesis for concurrent and distributed processing. AIAA J. **41**(10), 1996–2003 (2003)