Graph-based algorithms and data-driven documents for formulation and visualization of large MDO systems

Aigner, Benedikt; van Gent, Imco; La Rocca, Gianfranco; Stumpf, Eike; Veldhuis, Leo L.M.

DOI
10.1007/s13272-018-0312-5

Publication date
2018

Document Version
Final published version

Published in
CEAS Aeronautical Journal

CEAS Aeronautical Journal (2018) 9:695–709
https://doi.org/10.1007/s13272-018-0312-5

ORIGINAL PAPER

Graph-based algorithms and data-driven documents for formulation and visualization of large MDO systems

Benedikt Aigner[1] · Imco van Gent[2] · Gianfranco La Rocca[2] · Eike Stumpf[1] · Leo L. M. Veldhuis[2]

Received: 11 January 2018 / Revised: 14 May 2018 / Accepted: 25 May 2018 / Published online: 20 June 2018

Abstract
A new system is presented that enables the visualization of large multidisciplinary design optimization (MDO) problems and their solution strategy. It was developed within the scope of the European project AGILE. In AGILE, collaborative MDO is performed in large, heterogeneous teams of experts by solving MDO problems using a collection of design and analysis tools. This paper focuses on the visualizations required to support the formulation phase of an MDO project. The Knowledge and graph-based AGILE Design for Multidisciplinary Optimization System (KADMOS), an open-source MDO support system developed by Delft University of Technology, uses graph-based analysis to formulate an MDO problem and its solution strategy, based on the disciplinary analyses available in a repository. The results of KADMOS are stored in the standardized format CMDOWS (Common MDO Workflow Schema), which comprises the entire information on an MDO system. Although, based on Extensible Markup Language, the readability of the CMDOWS file is quite poor also for MDO experts, especially for large MDO systems involving thousands of variables. Providing visualization capabilities to thoroughly inspect the outcome of the different MDO formulation steps becomes a key factor to enable the specification of large MDO systems in a heterogeneous team. Therefore, VISTOMS (VISualization TOol for MDO Systems), a dynamic visualization package, was developed by RWTH Aachen University to enable the visualization and inspection of the different MDO system specification steps, thereby removing one of the main hurdles for using MDO as a development process. The developed visualization capabilities are demonstrated by means of an aerostructural wing design optimization project.

Keywords MDO · Visualization · KADMOS · CMDOWS · VISTOMS

Abbreviations

AGILE	Aircraft Third-Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts
MDAO	Multidisciplinary design analysis and optimization
CPACS	Common parametric aircraft configuration schema
MDG	MDAO data graph
CMDOWS	Common MDO workflow schema
MPG	MDAO process graph
CSS	Cascading style sheets
RCE	Remote component environment
CSV	Comma-separated values
RCG	Repository connectivity graph
DOE	Design of experiments
REMS	Reconfigurability in MDO problem synthesis
FDT	Functional dependency table
SVG	Scalable vector graphics
FPG	Fundamental problem graph
VISTOMS	VISualization TOol for MDO Systems
HTML	Hypertext markup language
XDSM	Extended design structure matrix
JSON	JavaScript object notation
XML	Extensible markup language
KADMOS	Knowledge and graph-based AGILE design for multidisciplinary optimization system

✉ Benedikt Aigner
aigner@ilr.rwth-aachen.de

[1] Institute of Aerospace Systems (ILR), RWTH Aachen University, Wuellnerstr. 7, 52062 Aachen, Germany

[2] Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1 2629 HS Delft, The Netherlands

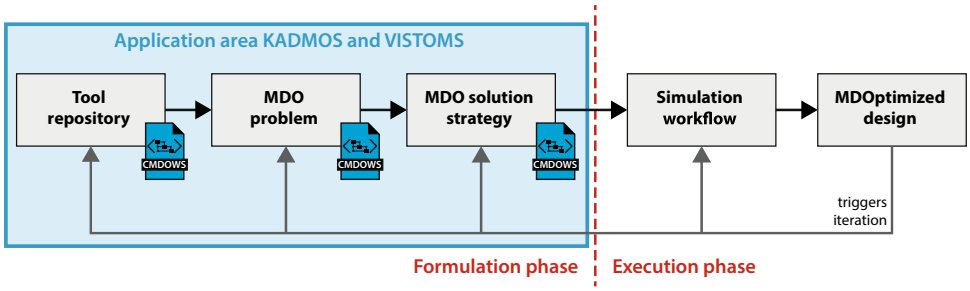

Fig. 1 Overview of the MDO development process and its two phases

1 Introduction

Past research indicates that MDO can offer huge benefits in complex product design. Boeing Phantom Works' scientists [6, 30] estimate that MDO can offer 8–10% gains for innovative aircraft design and even 40–50% gains for designing radically new and undeveloped concepts [17]. Despite the high potential gains, MDO is not as widely used as one would expect. Both technical [1] and non-technical barriers are hampering its full exploitation, as discussed below in this section [4, 27, 28].

To get a better understanding of the scope of the work presented here, it is convenient to refer to Fig. 1, where the different parts of the MDO development process are illustrated. The MDO development process in this figure can be roughly cut in half, with the formulation phase on the left side and the execution phase on the right. In the formulation phase, the tool repository is defined (or provided), the MDO problem to be solved is formulated, and a formal specification of the MDO solution strategy used to solve the problem is defined. This inexecutable MDO solution strategy is the blueprint of the executable workflow. The actual, executable MDO workflow is created in a simulation workflow platform of choice (e.g., RCE,[1] Optimus[2]) and run to find the optimal design. In a realistic design situation, the optimization is not performed just once; rather, the analysis of the design that is found after a certain run will provide new insights. These insights will be translated to an adjustment of the MDO problem formulation (e.g., change of objective, addition of constraints, etc.) and a reconfiguration of the associated MDO solution strategy (e.g., addition, removal, and replacement of analysis tools). This process of problem adjustment and process reconfiguration is iterated until a satisfactory design is found, or the project deadline has been reached. The focus of the system development described in this paper is on the visualization of the blocks in the formulation phase: tool repository, MDO problem, and MDO solution strategy.

One of the most critical technical barriers for MDO comes from the large (and continuously increasing) size of typical MDO problems. In the words of Pate et al. [23], the formulation of these problems has become increasingly complex as the number of analysis tools and design variables included in typical studies has grown. In this context, the problem of determining a feasible data flow between tools to produce a specified set of system-level outputs is combinatorially challenging. Especially, when complex and high-fidelity tools need to be included, the cost and time requirements to integrate the MDO system can easily approach the cost and time requirements of creating any of the discipline analyses themselves.

These cost and time requirements for the integration of the MDO system have also been identified in several research projects that have attempted to perform MDO by automating a full chain of design tools. In the previous projects of the DLR, it was found that the majority of the project time (60–80%) [7] would be used to create such an automated chain for aircraft design tools. Similar conclusions were drawn by Flager and Haymaker [8] who performed research into the design process metrics of both a legacy (current) design method and an MDO development process for the design of a hypersonic vehicle by Boeing [6, 30], see Fig. 2. In this figure, it is clear that the set-up time of the MDO workflow exceeds the 6 week set-up time of

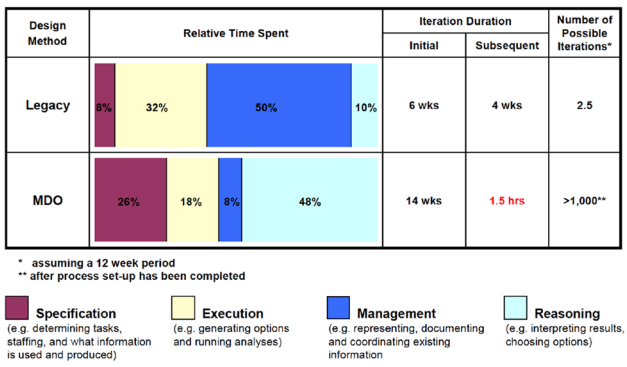

Fig. 2 Comparison of legacy design and MDO development process metrics for the design of a hypersonic vehicle [8]

[1] http://rcenvironment.de/.

[2] https://www.noesissolutions.com/our-products/optimus.

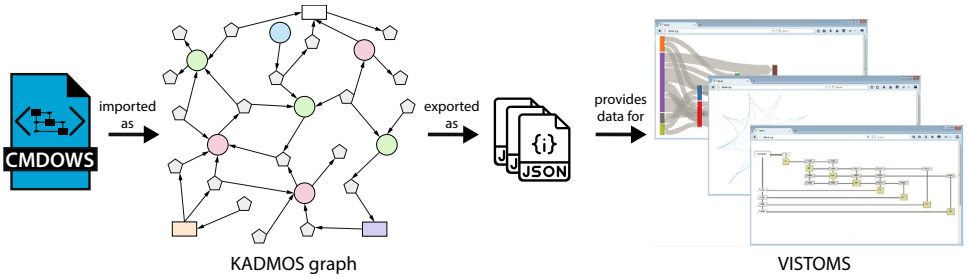

Fig. 3 Top-level overview of the dynamic visualization approach

the legacy method by 133%. This figure also shows that with the MDO design method, one needs to spend more resources in the 'Specification' phase category. This is to be expected, since, during the set-up of a fully automated chain of design tools, one needs to know down to the smallest detail what information is used and produced by each tool and which data are fed back and forward within a certain tool execution sequence. If the specification of the individual design tools and the overall MDO system could be improved, then the set-up time of the MDO process can be drastically reduced as well, thereby making the MDO approach even more convenient with respect to the conventional approach.

It is our conviction that the cost and time requirements for the integration of tools in a large and complex MDO system could be reduced by enabling the designer to analyze, visualize, and inspect the MDO system down to the smallest detail during the integration effort. Any system integrator is aware of the value of such analysis and visualization, but, in practice, the manual generation and update of this sort of documentation is too cumbersome. For example, if visualizations of the tool repository or MDO solution strategy are created, they are usually a collection of spreadsheets and text documents, which have to be updated manually and are hard to keep consistent. On top of that, such manually created overviews are not (fully) machine-readable and thereby cannot be used for any further automated analysis, integration, or manipulation of the MDO system. Therefore, the analysis, visualization, and inspection of MDO systems should be automated and based on machine-readable files or documents.

This is one of the main goals of the EU project AGILE,[3] where the developments described in this paper are taking place. The process for the generation of the necessary visualizations to support the MDO formulation phase is based on the outcome of KADMOS, the Knowledge- and graph-based AGILE Design for Multidisciplinary Optimization System developed at DUT.[4] KADMOS takes care of the automatic integration of the various design and analysis tools in the MDO system and supports the formulation of the MDO problem at hand and its solution strategy. KADMOS' functionalities are briefly explained in Sect. 2.2, while detailed information can be found in another publication [12]. As illustrated in Fig. 1, KADMOS stores the output of the MDO formulation process by means of a standardized XML format, called CMDOWS (Common MDO Workflow Schema), which is discussed in Sect. 2.1. The creation of the visualizations is done by coupling the CMDOWS files to a custom-built visualization package developed at RWTH Aachen University. This system, called VISTOMS (Visualization Tool for MDO Systems), is the main subject of this paper and its functionalities are described in detail in Sect. 2.3. The produced visualizations are presented in Sect. 3, based on a real MDO system for wing optimization, created within the AGILE project.

2 Methodology

As mentioned, the developments presented in this paper are based on two software packages: the MDO system formulation tool KADMOS and the visualization package VISTOMS. A top-level overview of the collaboration between KADMOS and VISTOMS is shown in Fig. 3. The approach is referred to as the 'dynamic visualization approach', where dynamic refers to the ability of interactive visualization objects to change appearance under mouse 'hovering' and clicking (more in Sect. 2.3). The set-up has been done, such that the graph information for any of the first three blocks in Fig 1 (which is usually stored in CMDOWS files) is translated by KADMOS into the JSON (JavaScript Object Notation) representation required by VISTOMS. This collection of JSON data is then visualized with VISTOMS through an HTML (Hypertext Markup Language) page that can be opened in any web browser and includes the interactive visualization objects. It should be noted that the use of KADMOS shown in Fig. 3 only represents a small part of the package. The majority of KADMOS is actually geared towards producing and editing the CMDOWS files (more in Sect. 2.2), not at the postprocessing for which it is used

[3] Aircraft Third-Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts, see: http://www.agile-project.eu.

[4] See: https://bitbucket.org/imcovangent/kadmos.

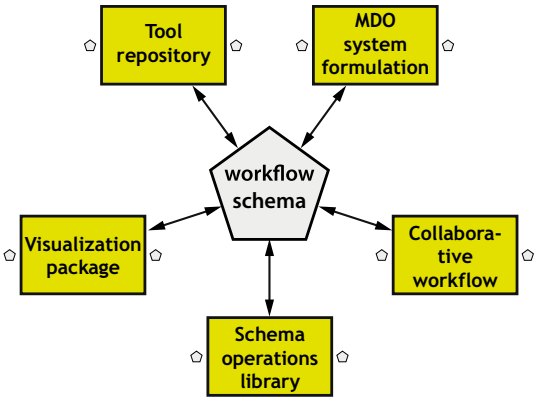

Fig. 4 Concept of exchangeability between different MDO framework applications through a workflow schema [11]

in the dynamic visualization approach. All elements of the top-level overview in Fig. 3 are discussed in the upcoming section.

2.1 CMDOWS

CMDOWS is an open-source,[5] XML-based workflow schema that was developed at DUT to enable the exchange of the formulated MDO system between MDO framework applications, as visualized in Fig. 4. Other publications have already addressed the deployment of a formalized schema as an exchange format for MDO applications [14–16]. However, for the purpose of the AGILE project, it was necessary to use something that is feasible for collaborative MDO projects and connecting a wider range of MDO applications. Therefore, CMDOWS was created within the project as a new schema, as described in a publication by Van Gent et al. [11]. The different stages of the MDO system in the formulation phase can all be stored in the CMDOWS format. Each stage in Fig. 1 (from left to right) enriches the CMDOWS file to go from a repository of design tools to a full description of the optimization strategy. KADMOS is able to provide the graph-based representation for each stage (see next section) and can store these in a CMDOWS file; however, visualizing the outcome of these stages is not properly handled by KADMOS. In addition, the CMDOWS files, although based on XML, do not offer a practical means to the user for inspecting the generated MDO system formulation. Therefore, the work presented here was focused on the link between the visualization package VISTOMS and CMDOWS. Within that link, the efficient graph-based algorithms of KADMOS were used for data processing purposes.

[5] Available at: http://cmdows-repo.agile-project.eu.

2.2 KADMOS

As mentioned in the previous section, the use of KADMOS is twofold in this work:

- Creation of the CMDOWS files to be used as input for the approach in Fig. 3
- Graph-based data processing of the CMDOWS files to convert the XML representation into the JSON format required for VISTOMS

Both applications of KADMOS are only discussed briefly here.

2.2.1 KADMOS as MDO system formulator

The ability of KADMOS to support the specification of the MDO system is discussed in detail in earlier work [12]. A mapping between the three stages of the formulation phase in Fig. 1 and the associated KADMOS graphs is shown in Fig. 5. Four different graph types are associated with the three formulation phases.

The repository connectivity graph (RCG) is an object that represents the design and analysis tool repository as a web of data containing function and variable nodes and their connections. This graph is established easily for large tool databases by exploiting the central data schema approach, such as CPACS [22]. The example in Fig. 5 (left) concerns a very small repository with only eight function nodes (design competences) and ten variables. Larger tool repositories, such as the one used in the results section of this paper, are still stored as a graph structure, but their visualizations, as expected, have severe readability limitations.

The MDO problem is represented in KADMOS with the fundamental problem graph (FPG), see Fig. 5 (middle). The FPG is a subset of the RCG in terms of nodes and edges. In addition, its nodes are also enriched with attributes required to specify the MDO problem at hand, such as design variables, objective, and constraints.

Finally, the neutral representation of the MDO solution strategy is stored in two separate graph constructs: the MDAO process graph (MPG) and the MDAO data graph (MDG), where the first contains the process execution flow of the various MDO system components, and the second specifies the specific data exchanged between those components. These graphs are created automatically by KADMOS based on the FPG and stored in the same CMDOWS file, thereby using all the elements of the schema.

2.2.2 KADMOS as data processor

The system formulation capabilities of KADMOS cover all stages of the formulation phase given in Fig. 1, and can store

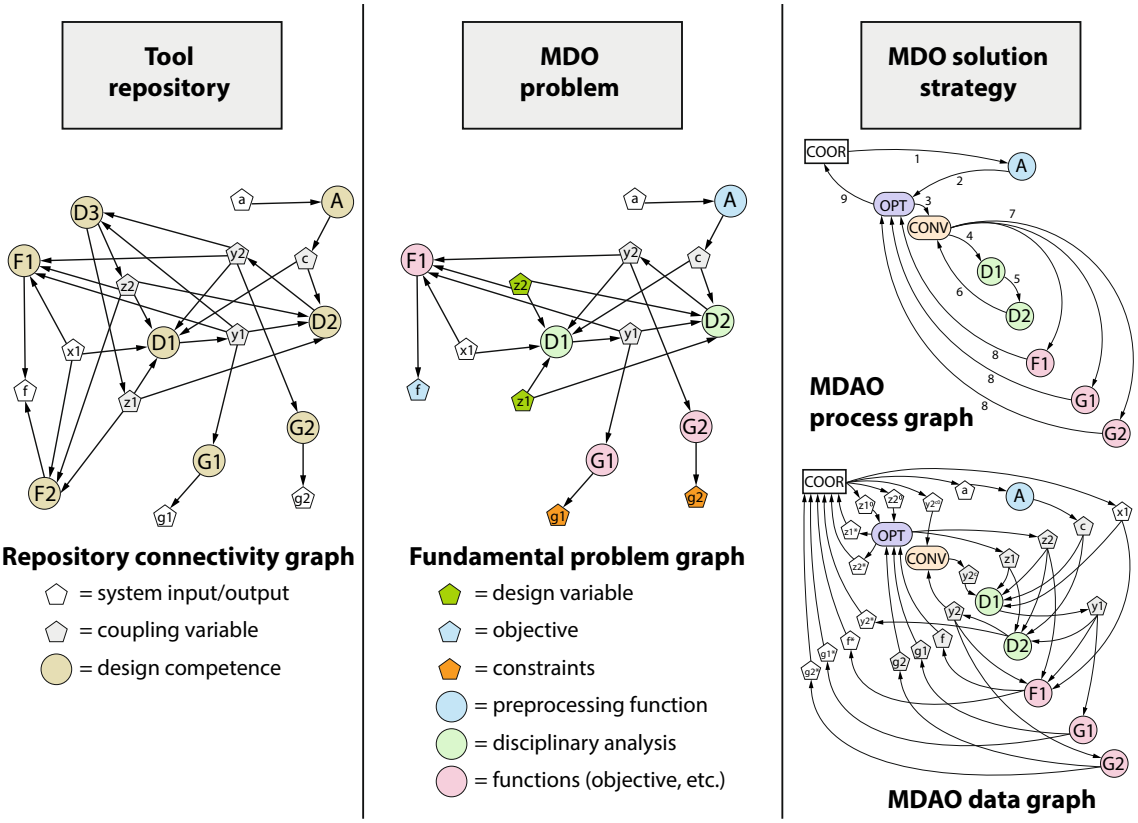

Fig. 5 Top-level overview of KADMOS and its relation to the formulation phase of the MDO development process in Fig. 1 (all visualizations are based on the Sellar problem [26])

the CMDOWS file for each stage, although these capabilities could be taken over by other platforms, as well. Especially, the creation of the tool repository is a relatively easy task that can be performed with other applications. For example, in AGILE, the business process platform KE Chain[6] [10] also contains a module to create a tool repository and export it as a CMDOWS file. However, even if other platforms create the CMDOWS file, KADMOS is still required as a data processor to provide the right data format for VISTOMS.

This data processing is required to provide VISTOMS with files that are directly interpretable; hence, no cumbersome analysis of the data is required before visualizing it. In other words, the KADMOS graphs do contain all the information that is required to visualize it, but some of the information is stored implicitly. To improve responsiveness of the visualization package, this information is transferred explicitly to the JSON files read by VISTOMS. An example of this would be the input and output variables of a single tool. This information is stored in the graph, but, to determine this information for a single tool, one has to loop over all the incoming and outgoing connections of the tool. Instead, the input and output variables per tool are stored explicitly in the JSON files, so that VISTOMS does not have to perform the loop when the information is requested. Similarly, if the variables follow a central data schema, then the hierarchy of the variables (which is lost in the graph representation) is reestablished based on the variable names and stored in a nested dictionary in the JSON files.

The KADMOS data processing step provides VISTOMS with easily accessible information about the MDO system to be visualized. The processing, which takes in the order of seconds (depending on the size of the system), prevents a lot of waiting time when using the dynamic visualizations. Downside of the JSON files with directly useable information is that some information is stored multiple times in slightly different ways, thereby increasing the size of the collection of JSON files with respect to the original CMDOWS file. However, this decrease in storage efficiency is well worth the associated performance increase when using the dynamic visualizations.

2.3 VISTOMS

The combination of KADMOS itself and the visualizations developed in the course of this research provide the MDO

[6] See: https://www.ke-chain.com.

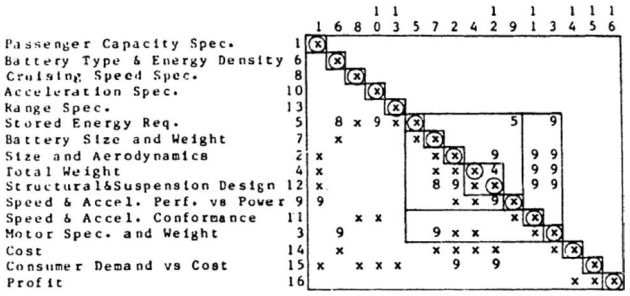

Fig. 6 Design structure matrix (DSM) [29]

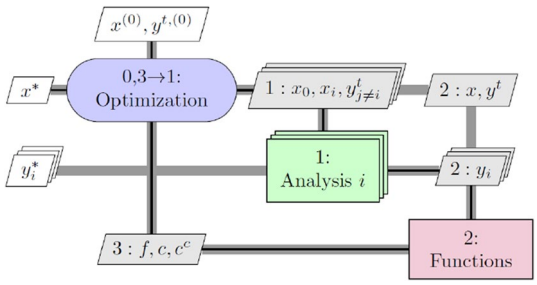

Fig. 7 XDSM for an individual discipline feasible (IDF) architecture [19]

integrator with a powerful set of tools to support creation, inspection, debugging, and modification of large and complex MDO problem formulations. The presented visualizations are obtained using the open-source library D3.js [5]. In the following sections, D3.js as well as significant state-of-the-art visualization techniques are currently used in MDO will be introduced. Subsequently, the newly developed visualization techniques embedded in VISTOMS will be presented. Their ultimate goal is to enhance the understanding of complex MDO systems, which is necessary for effective MDO problem formulation, documentation, and knowledge sharing.

2.3.1 D3.js library

D3.js is an open-source JavaScript library, developed and released by Bostock under the BSD license, for creating and modifying documents based on data [5]. D3.js supports the user in visualizing any data by combining the standards HTML, SVG (Scalable Vector Graphics), and CSS (Cascading Style Sheets). It is, therefore, an easy to use and powerful tool for visualization that can be opened with any standard web browser. The coding can be directly performed within an HTML file, which can then be opened in the web browser showing the embedded visualizations. The library comes with a predefined set of standard visualization techniques that can be easily accessed, modified, and extended. Any visualizations, also those that are not predefined in D3.js, can be obtained and modified using standard SVG commands. The data behind the visualizations can be stored in JSON (JavaScript Object Notation) or CSV (Comma-Separated Values) files, and are accessible via JavaScript code.

2.3.2 State-of-the-art visualization techniques in MDO

In the field of MDO, the visualization of MDO systems is widely recognized as a valuable tool to enhance knowledge about the problem formulation at hand. Therefore, over the years, various visualization techniques have been developed. The N2 chart, introduced by Lano in 1977 [20], is, for instance, a well-known method for visualizing system couplings. The Design Structure Matrix (DSM), which is similar to the N2 chart, was developed by Steward in 1981 [29] and shows inter-dependencies between competences in a square adjacency matrix (see Fig. 6).

In Fig. 6, each of the non-blank off-diagonal elements (x's and numbers) represents a data dependence between the competences, which are arranged on the diagonal. Both DSM and N2 chart thereby enable the representation of the data exchanged among the various competences by showing the data dependence in a system (see bottom right graph in Fig. 5). However, these visualizations are not effective in formalizing the execution order of the tools and the triggering of the various loops including any required iterations by convergers or optimizers. However, with increasing number and complexity of analysis tools, the choice of competence execution order becomes more important and more complex, as well. Wagner and Palambros developed the so-called functional dependence table (FDT) to account for constraints and objectives in an MDO problem formulation [31]. The drawback of the FDT is that information about inter-dependencies between competences and functions is partially lost, and therefore, a competence execution order cannot be indicated. A combination between DSM and FDT called Reconfigurability in MDO Problem Synthesis (REMS) was introduced by Alexandrov and Lewis enabling indication of couplings between competences as well as constraints and objectives [2]. Nevertheless, the execution order of the competences is not available within the representation of REMS.

This capability is enabled by the extended DSM (XDSM) introduced by Lambe and Martins in 2012 (see Fig. 7) [19].

The XDSM provides a visualization that captures the full description of an MDO problem, combining the advantages of DSM and FDT. In general, an XDSM can be read as a square adjacency matrix, where the competences are arranged on the diagonal and the columns and lines indicate competence inputs and outputs, respectively. The competences are connected via data pipelines (gray lines) indicating data transfer. Feed-forward connections

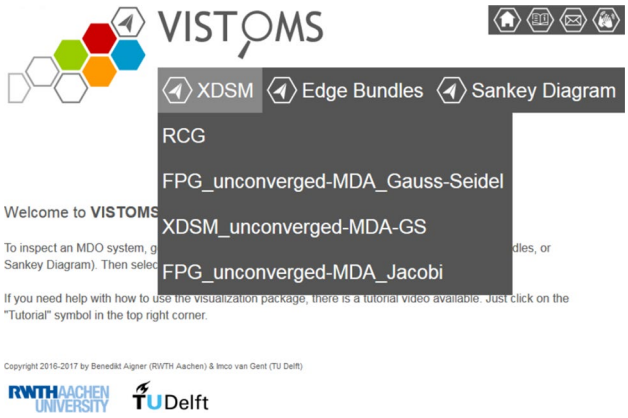

Fig. 8 Main page of VISTOMS. The different graphs of the MDO system (RCG, FPG, etc.) can be selected via drop-down menu using any of the three visualization techniques (XDSM, Edge Bundles, and Sankey Diagram)

are shown on the right and feedback connections on the left side of the diagonal. The so-called edges (rhomboids on the off-diagonal) indicate a connection between two competences (also referred to as couplings), i.e., the information that is processed from one competence to the other. The so-called process lines (thin black lines) in combination with the numbers in the diagonal blocks indicate the order of the workflow execution (MDAO process graph, c.f., Fig. 5).

Although the XDSM offers the means for a detailed and comprehensive description of an MDO system, its readability quickly degrades when the number of competences and their coupling increases, at least in its static document-based version.

The HTML-based rendering approach of the presented visualization package enables the use of effective standard representations, such as the XDSM, while offering the dynamic scaling and displaying options necessary to guarantee readability and inspectability also for MDO systems of extremely large size. Examples are discussed in the next section.

2.3.3 Visualization techniques for CMDOWS files

For the CMDOWS files, three main visualization types have been selected and further developed:

- XDSM;
- Egde Bundling View;
- Sankey Diagram.

The visualization package is accessible via web browser. Figure 8 shows the starting page of VISTOMS.

Note that VISTOMS provides solely a visual representation of an MDO system, in which no actual competences can be executed. Rather, the automated creation of executable workflows from the MDO architecture is a task, which is performed in simulation workflow platforms by parsing a CMDOWS file [13].

In the following sections, the three above-mentioned visualizations will be described in detail with respect to their capability to enhance the insight into MDO systems. The interested reader can directly access and experience a number of example visualizations via the open-access CMDOWS browser interface.[7] On the browser interface, a number of pre-generated CMDOWS files are available for demo purposes. Note that the VISTOMS visualizations can also be created for any MDO system using the open-source KADMOS package.

XDSM The enhanced XDSM visualization developed in this research is based on the open-source XDSMjs package, which was released by Lafage in 2016 [9, 18]. The main structure is the same as the conventional XDSM (see Fig. 7), while the major difference between the two is that, using the D3.js library, the XDSM can be accessed dynamically and interactively via a web browser. However, for large and complex MDO systems, not all the embedded information can be clearly visualized at once with the XDSMjs package. Therefore, within the scope of the presented research, the XDSMjs package was further enhanced to give the user the possibility to access the full information embedded into an MDO system in a human intelligible way. While the basic layout was kept simple, more detailed information can be inspected interactively on demand. The main features of the XDSM view in VISTOMS are given in Fig. 9 showing an MDO system architecture for the Sellar problem,[8] which was already presented in Sect. 2.2 (see also Fig. 5 for the KADMOS graph representations of this MDO problem).

Note that the overlay frames in Fig. 9 are not visible in the actual visualization package and have only been included for this paper to explain the visualization capabilities. For this purpose, the focus of Fig. 9 is set on the connection between the competences *DOE* and *D1*. For detailed inspection of the XDSM, the user has several options. Hovering over an edge with the mouse displays the names of the underlying data that are processed here. Right clicking on an edge gives the user two options. First, it is possible to examine basic information about the edge such as the total number of connections and their dimension. Second, the user can further examine the

[7] Available at: http://cmdows.agile-project.eu.

[8] VISTOMS for Sellar problem available at: https://www.agile-project.eu/files/VISTOMS_SellarProblem.

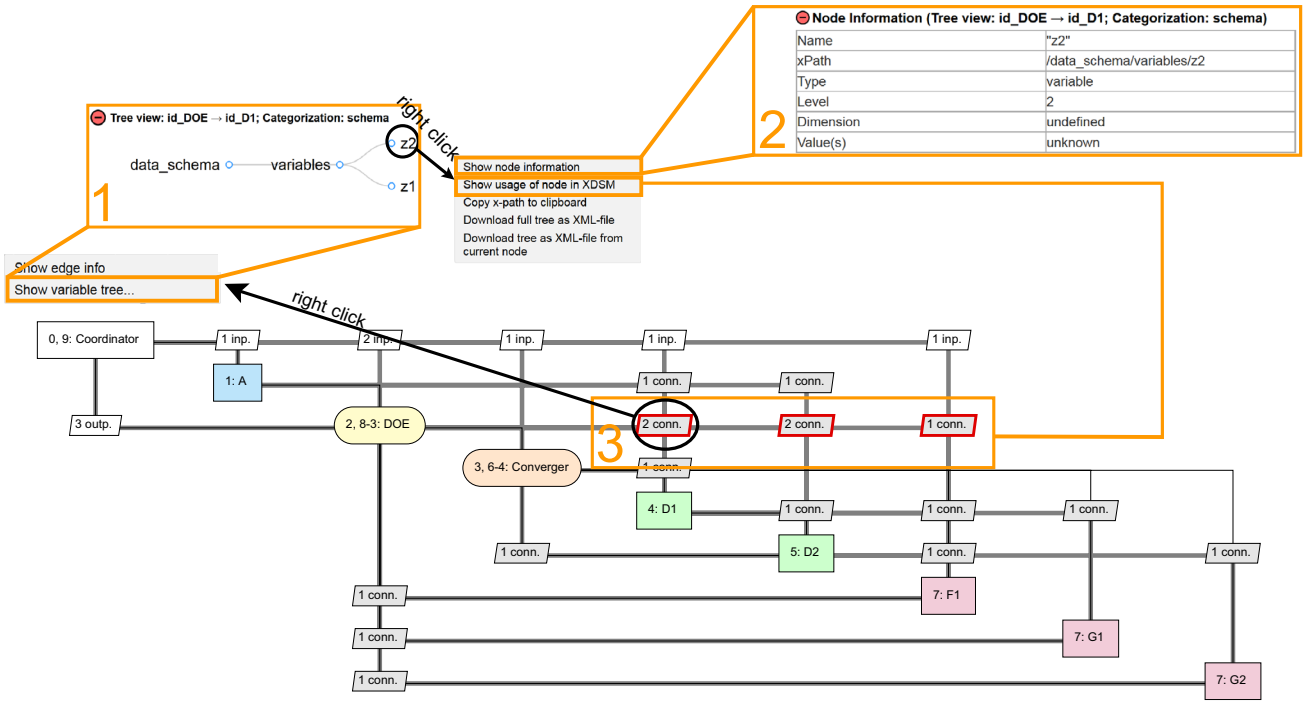

Fig. 9 VISTOMS XDSM for the Sellar problem with a converged Gauss–Seidel design of experiments (DOE)

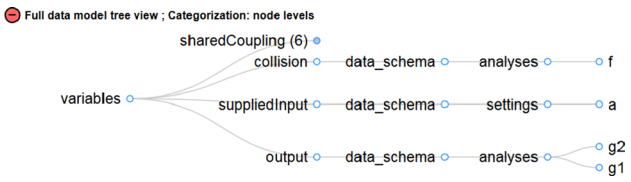

Fig. 10 Tree layout according to node levels, Sellar problem DOE

underlying data of the selected edge. The data are shown as a hierarchical tree [5] containing a subset of the underlying data model (e.g. CPACS schema as an XML-based parameterization of an aircraft) where the categories and subcategories are represented by the branches and leafs (see Fig. 9, overlay frame 1). The tree view is expandable and collapsible via mouse click according to the user's requirements. The layout can be organized according to different categorizations. These include the basic hierarchical data schema (e.g. CPACS), but also, for instance, a categorization according to the node types in the MDO system (see Fig. 10).

The latter can, for instance, be helpful, when multiple competences modify the same variable. This so-called collision can potentially cause problems such as inconsistencies in the MDO system and, therefore, needs to be at least recognized by an MDO integrator. Note that the tree layout shown in Fig. 10 is not fully expanded to the last leaf nodes, the node *sharedCoupling* is collapsed, while it is indicated in the brackets that there are six leaf nodes contained here. This feature gives the user an idea on how many variables are contained in the layout, even when the tree is not fully expanded, and, therefore, keeps the layout clear, which is especially required for large data sets.

Each of the nodes in the tree layout can be further examined via right click. In the example given in Fig. 9, the selected node of interest is variable *z2*. Several options for examination exist, such as indication of general information about a node (name, type, dimension, or its current value), as can be seen in Fig. 9, overlay frame 2. Another option is to display the occurrence/usage of a node in the MDO system. This means that it can be highlighted, wherein the MDO system a node is processed from one competence to another by highlighting the respective edges (see Fig. 9, overlay frame 3). This option provides valuable information when setting up a problem solution, because the MDO integrator can easily examine how the competences are connected to each other and which of the processed variables are of most interest due to their occurrence in the system. Thus, an overview on whether the competences are connected correctly, or at least as expected, is given. Furthermore, it is possible to download the tree layout as an XML file (including current values of leaf nodes) to, for example, manually adjust the data set or to simply extract the data from the visualization. These dynamic, interactive inspection possibilities are the major advantage of the presented visualization package and

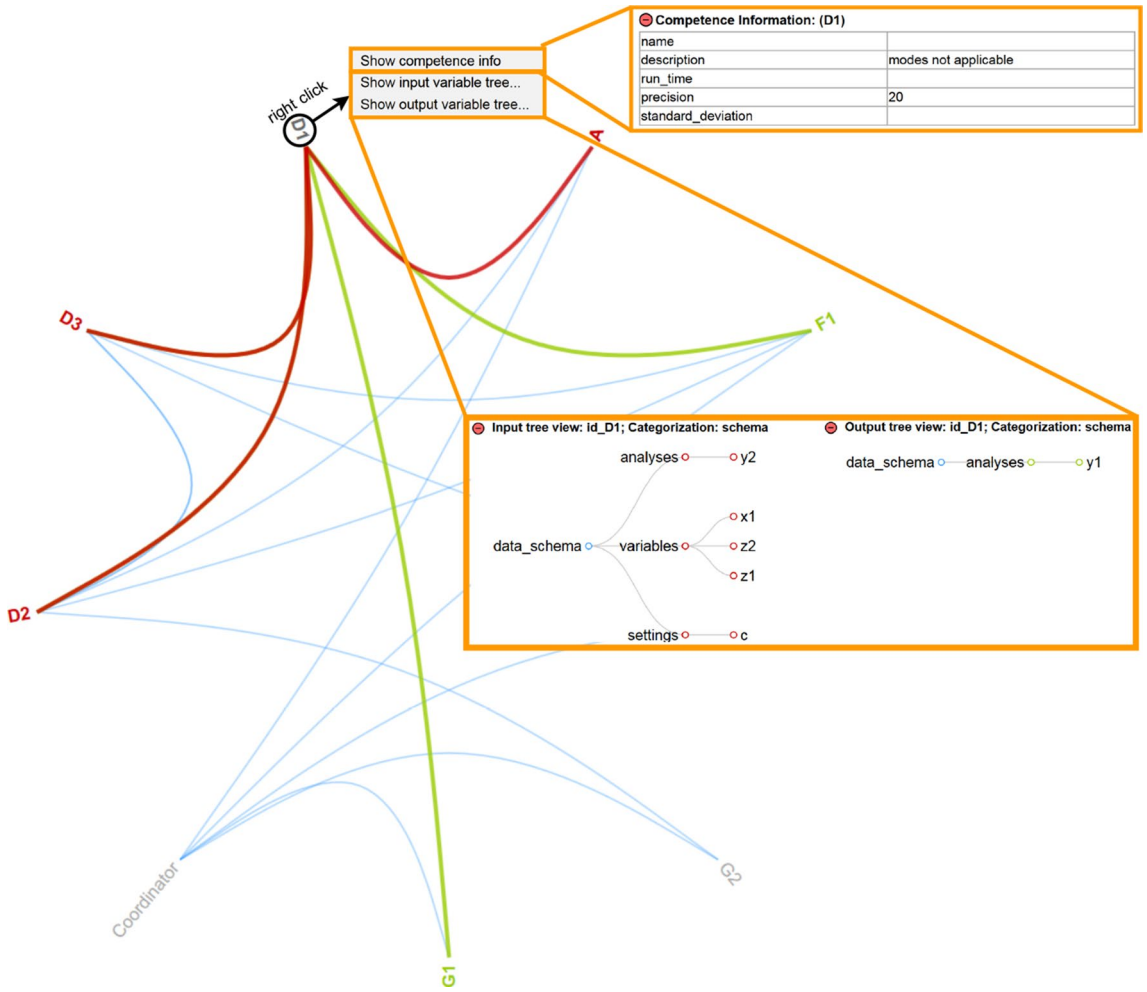

Fig. 11 VISTOMS Edge Bundling view for the Sellar problem RCG

make it a convenient debugging tool for MDO systems of arbitrary size and complexity.

The above described techniques, developed in the course of this research, are quite similar for all three of the visualizations (XDSM, Edge Bundling View, and Sankey Diagram). Thereby, recurring visualization elements are established, which facilitates the usability of the functions.

Edge Bundling View The so-called Edge Bundling view is a circular layout of interconnected elements and is adapted from an example by Bostock [5]. The basic idea of this visualization is given in Fig. 11 with the example of the Sellar problem RCG.

Each of the blue lines indicates a general dependence between two elements. The focus can be set on any element by hovering over it with the cursor (c.f. element *D1* in Fig. 11). The red lines indicate input flow to the element and the green lines indicate output flow from the element. Further detailed inspections can be carried out via right click on the connecting lines or on the elements, similar to what has been described for the XDSM in the previous section.

In contrast to the XDSM view, the Edge Bundling view only provides data information, i.e., the interconnections of the competences and the data processed between them, whereas the workflow process information cannot be displayed. On the other hand, it visualizes the connections among the competences more intuitively.

Sankey Diagram The Sankey Diagram was first introduced by Henry Riall Sankey in 1896 for the visualization of energy flows in steam engines [24, 25]. A variation of the conventional layout is the bi-directional Sankey Diagram, which was used in a D3.js-based package by Atkinson in 2015 [3]. While the conventional layout only considers one flow direction, the bi-directional layout is able to account for feed-forward and feedback information flow between the elements at the same time. The Sankey Diagram used in the presented research is based on the developments by

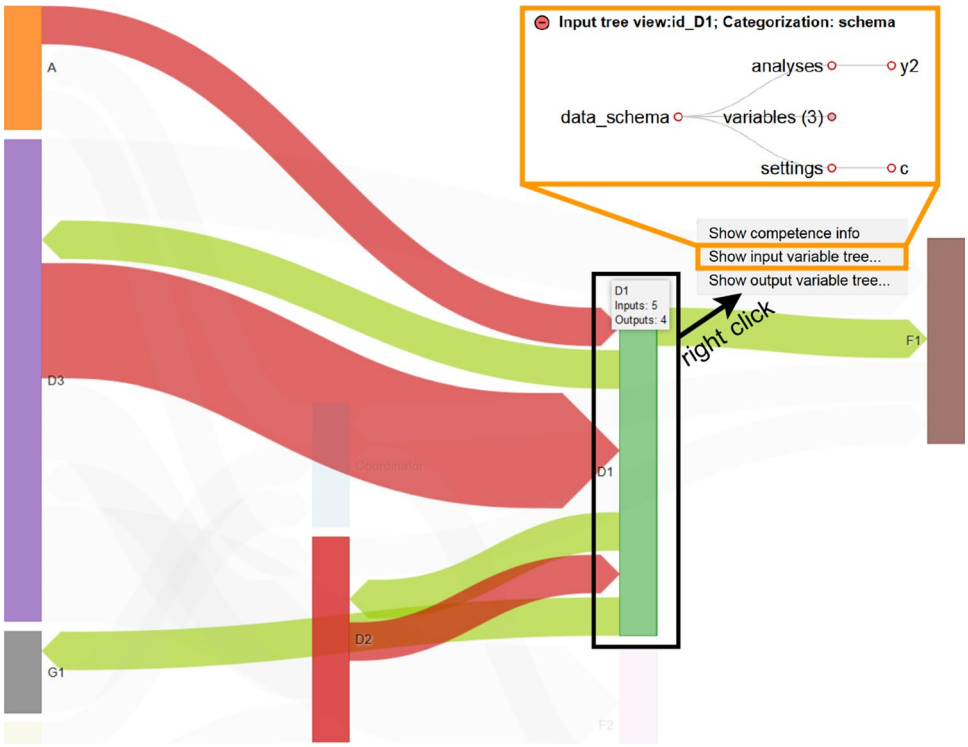

Fig. 12 VISTOMS Sankey diagram for the Sellar problem RCG

Atkinson. An example of the visualization for the Sellar problem RCG is shown in Fig. 12.

Again, for any competence in the system, input information flow is displayed in red and output information flow is displayed in green. The width of the connecting arrows refers to the amount of information transferred between two elements. A connection with 100 variables transferred from one element to another will appear wider than one with only ten variables. As with the two other visualizations, it is possible to further analyze the graph with help of the previously described dynamic inspection capabilities.

3 Results

Within the scope of the AGILE project, the visualization package presented in Sect. 2.3 has already been intensively used and tested by multiple AGILE project partners for various MDO problems. The developed capabilities have proven to effectively assist the MDO integrator in the problem formulation process [21]. In this section, the results of the visualization enhancements will be presented with regard to a tool repository containing a collection of DUT aircraft design tools.[9] This use case concerns the aerostructural optimization of an aircraft wing. The starting point of this case is a tool repository containing over 28,000 variables and over 37,000 data connections based on 29 function nodes. More details on this case study can be found in [13].

XDSM Fig. 13 shows an extraction of the XDSM for the DUT wing design MDO system. Using the dynamic visualizations described in Sect. 2.3, the MDO system can be examined in detail. The focus of Fig. 13 is set on the two input edges of the competence *EMWET*, a tool that estimates the wing mass. A right click on the input edge coming from the competence *HANGAR[AGILE_DC1_L0_MDA]*, which provides an initial parameterized data set of the aircraft, leads to a hierarchical tree layout containing 167 pipeline variables as a subset of the CPACS data schema (Fig. 13, overlay frame 1). The expanded tree layout indicates that, for instance, the geometry of the main wing (*wing[mainWing_wingID]* is passed from *HANGAR[AGILE_DC1_L0_MDA]* to *EMWET*, which seems plausible, as the wing mass is strongly affected by its geometry. A right click on the wing geometry node enables a detailed examination of the node characteristics as displayed in overlay frame 2. Displaying the usage of the node, it can also be seen that the wing geometry is transferred at five other edges in this particular extraction of the XDSM (red-highlighted edges).

The other input edge (12 inp.) in the top of *EMWET* is provided by the so-called *Coordinator*, which represents the outer world and operates as the main control function of the MDO system (see Fig. 13, overlay frame 3). This

[9] VISTOMS for DUT wing design available at: https://www.agile-project.eu/files/VISTOMS_TUDWingDesign.

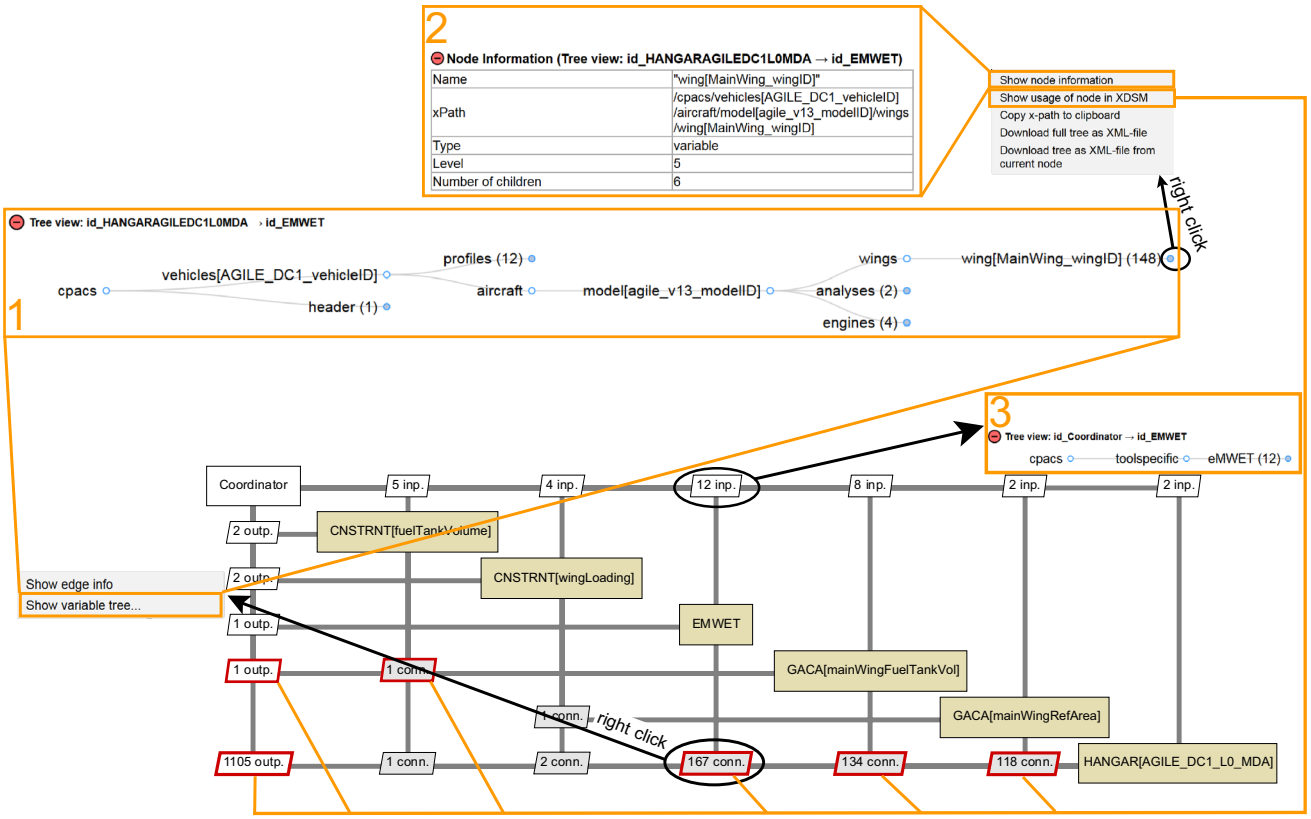

Fig. 13 VISTOMS XDSM for the DUT wing design RCG

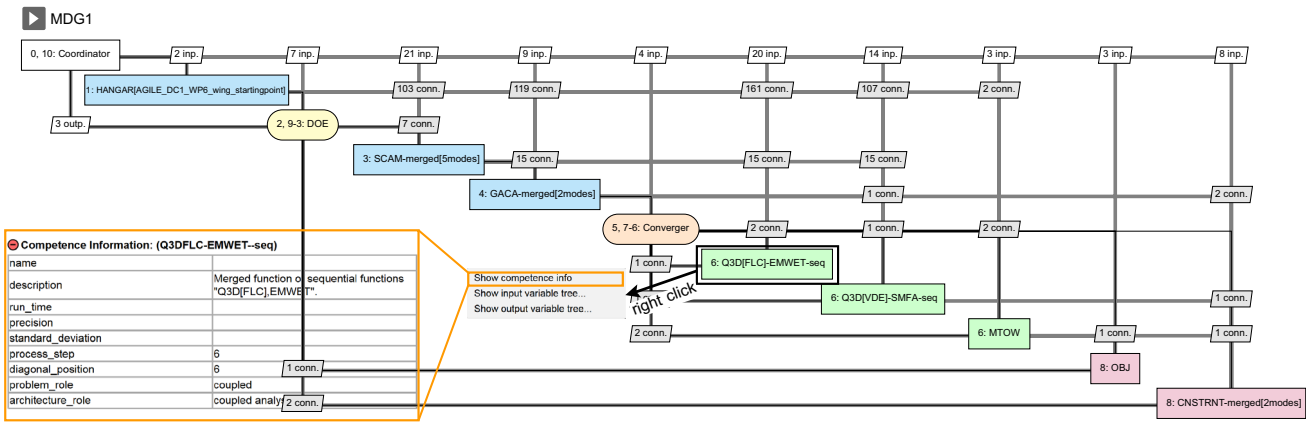

Fig. 14 XDSM of a DOE with Jacobi converger for the DUT wing design

means that every variable passed from the *Coordinator* to a competence is not provided by any other competence in the MDO system. For aircraft-specific information, such as the geometry or mass data, this should normally be avoided, because, in that case, the user would have to specify the values manually. In the presented MDO system, however, it can be seen that only tool-specific information, i.e. the tool settings of *EMWET*, are processed here. For an MDO integrator, this leads to the conclusion that *EMWET* receives all the required input data by the competences available in the tool repository. If this was not the case, the integrator would have to provide the missing information himself, or include an additional competence that can provide it.

As mentioned in Sect. 2.2, in the course of the MDO development process, the MDO system at hand becomes

more and more specified moving from a tool repository to an MDO problem and, finally, towards one or more MDO solution strategies (see Figs. 1, 5). Within the scope of AGILE, the XDSM view has proven to be a valuable visualization technique to grasp the full description of an MDO system. Especially, in the later phases of the MDO development process, it effectively combines the information of MDAO data and process graph (cf. Fig. 5) in one single view.

Figure 14 shows the XDSM view of an MDO solution strategy for the DUT wing design. The presented MDO architecture is a DOE with an internal Jacobi converger for the wing design use case. Compared with the XDSM for the RCG from Fig. 13, it can be seen that the amount of competences and data connections is narrowed down significantly for the specific use case. The coloring of the competences indicates their role in the MDO architecture (c.f. Fig. 5). The additional competences *DOE* and *Converger* have been integrated automatically by KADMOS. They are characteristic functions of this particular MDO architecture. Furthermore, in addition to the gray data pipelines connecting the competences (MDAO data graph), the process lines are now visible, which indicate the order of the workflow execution (MDAO process graph). These have also been included within the step of defining the MDO solution strategy by KADMOS. To give an example, the process lines going from the *Converger* to the competences *Q3D[FLC]-EMWET-seq*, *Q3D[VDE]-SMFA-seq*, and *MTOW* indicate that these three competences are executed in parallel, which is a specific characteristic of the Jacobi converger used in this MDO problem solution. As in any standard XDSM, this process information is also provided by means of the sequence number assigned to each diagonal block. In this case, the three aforementioned competences have all the same sequence number 6. The competence *Q3D[FLC]-EMWET-seq* is a combination of the sequentially running competences *Q3D* and *EMWET*, which have been merged into one competence block. *Q3D* performs an inviscid aerodynamic analysis to evaluate the aerodynamic loads on the wing, which are then used for the wing mass estimation by *EMWET*.

A right click on a competence gives the user the option to show the previously described hierarchical tree layout (see Fig. 13) of either all input or all output data for the respective competence. In addition, the user can be provided with competence information, including, e.g., its description, its role in the problem definition, or its role in the MDO architecture.

Within the process of re-configuring and adding new MDO problem solutions to an MDO case with KADMOS, the number of different MDO architectures for the same design problem can become quite large. One of the advantages of the presented visualization package is the fact that all of the graphs produced are eventually combined into one single package and can be selected via a drop-down list in the main menu of VISTOMS.

Edge bundling view In Fig. 15, the Edge Bundling view for the DUT wing design problem with a DOE Jacobi architecture is shown. As can be seen from the figure, the number and type of connections between the competences can be grasped very quickly in this view.

Therefore, within the scope of VISTOMS, the Edge Bundling view is very convenient to provide a general overview on interconnections between competences in an MDO system. It is noticeable that, for the tree view of an edge between two competences, one can see the connections in both directions at the same time (if applicable). Input nodes are marked as red circles, output nodes are marked as green circles, while the direction of the connection is given in the headline of the tree view (see overlay frame in the bottom of Fig. 15).

Sankey diagram An example of the Sankey diagram for the DUT wing design with DOE Jacobi architecture is shown in Fig. 16. As with the Edge Bundling view, the information of the MDO process graph is not provided in this visualization. Instead, the Sankey Diagram is a very convenient way to illustrate the magnitude of competence interconnections. To give an example, it becomes clear from Fig. 16 that the competence *Q3D[FLC]-EMWET-seq* is strongly coupled to the *HANGAR* tool, whereas the connection to the *Converger* is rather loose. This is plausible, because the *HANGAR* tool provides the full geometry description of the aircraft, while the convergence criteria of the *Converger* competence are only a small number of mass positions. Of course, to grasp the actual properties of the connections, these have to be examined in detail, which can be accomplished by the various inspection options of VISTOMS.

4 Conclusions

A new approach for the manipulation and visualization of MDO systems has been presented, which combines three major developments from the AGILE project:

- the standardized MDO workflow storage format CMDOWS
- the MDO problem formulation system KADMOS for automated machine-readable graph data
- the D3.js-based visualization package VISTOMS

The visualization capabilities have been demonstrated on an MDO system based on a realistic aerostructural wing design case to explain its set-up and show the ability to support an increased understanding of large MDO systems down to the smallest detail. The package reduces both technical and non-technical barriers of performing MDO. The

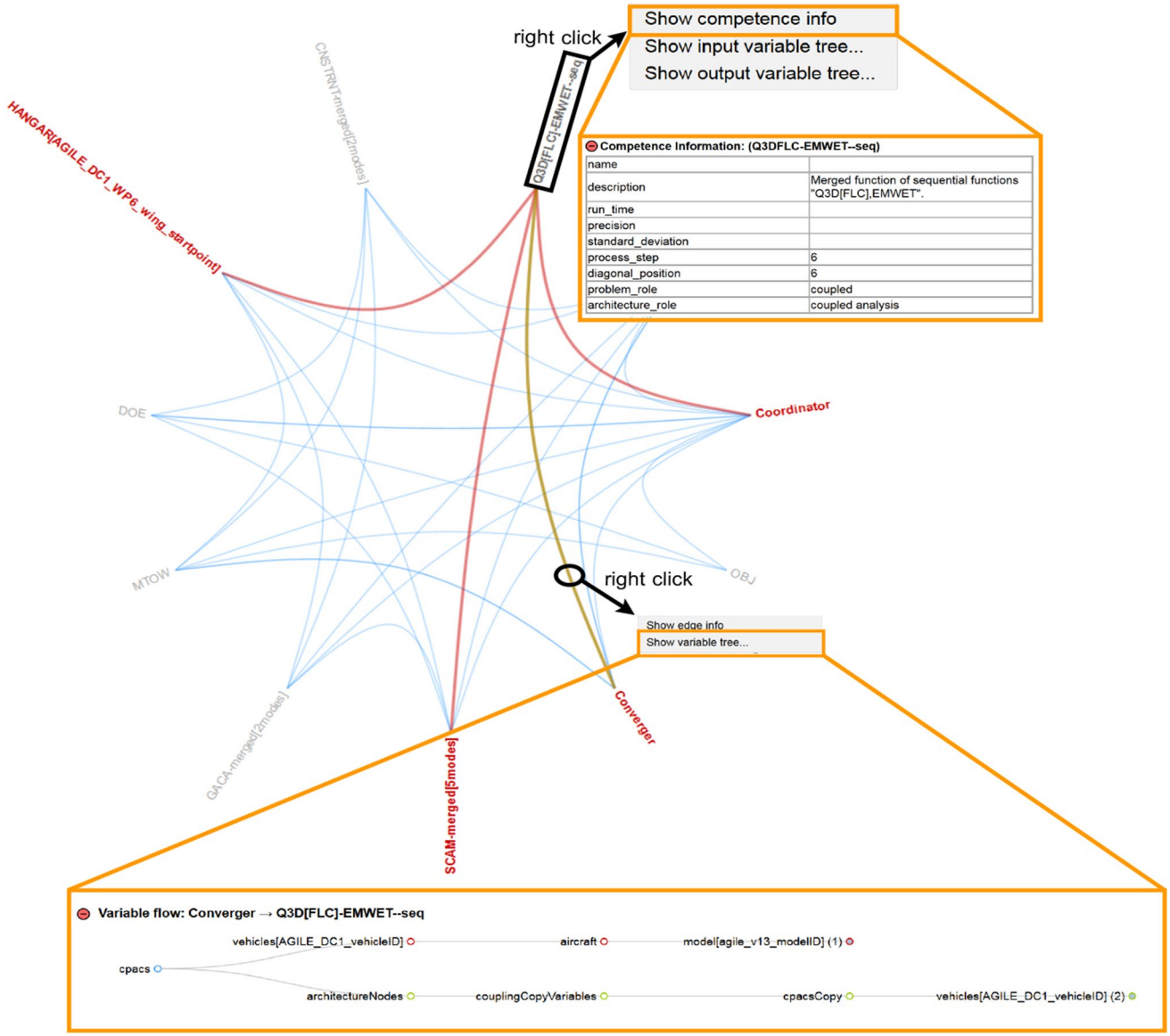

Fig. 15 Edge bundling view for the DOE Jacobi for the DUT wing design

technical barrier concerns the fully automated formalization and integration of large MDO problems with KADMOS. The non-technical barrier is tackled by the automatic, dynamic visualizations described in this paper that provide the opportunity to inspect, share, and document large MDO systems within heterogeneous design teams. Furthermore, the developments presented in this paper were embedded into the larger software architecture developed within the AGILE project called the AGILE framework, which is discussed in [10].

All developments presented in this paper are open source and form a fundamental output of the AGILE project. Future work for the open-source platforms will focus on extending their capabilities to handle a larger variety of MDO systems, as they will be put to the test in future AGILE design campaigns and other MDO projects.

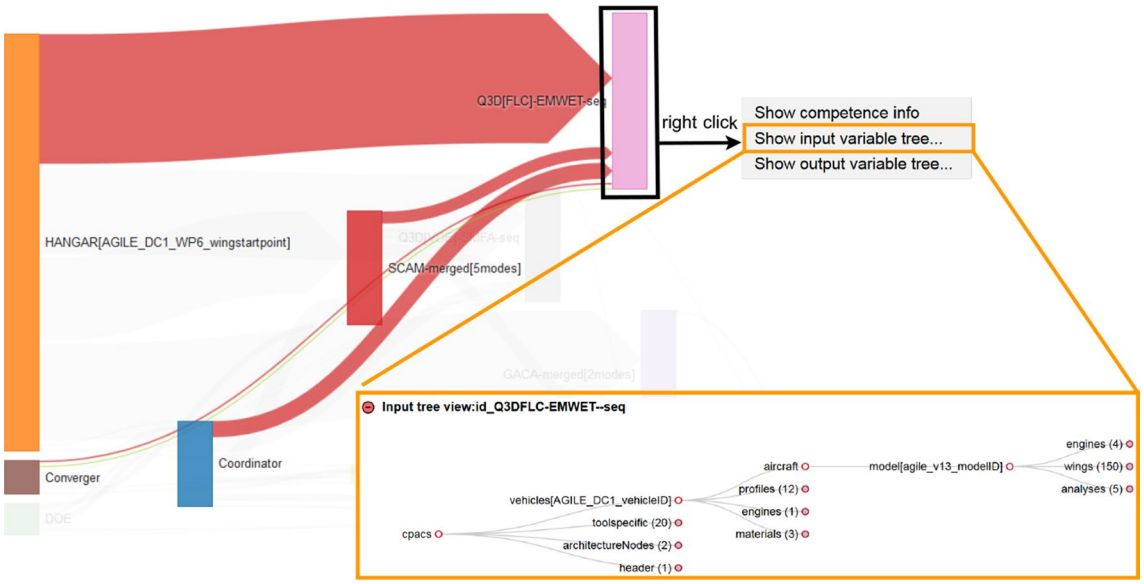

Fig. 16 Sankey diagram the for DOE Jacobi for the DUT wing design

Acknowledgements The research presented in this paper has been performed within the framework of the AGILE project (Aircraft Third-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.

Open-source references

CMDOWS repository	http://cmdows-repo.agile-project.eu
CMDOWS visualization interface	http://cmdows.agile-project.eu
KADMOS	https://bitbucket.org/imcovangent/kadmos
VISTOMS file for Sellar problem	https://www.agile-project.eu/files/VISTOMS_SellarProblem
VISTOMS file for DUT wing design	https://www.agile-project.eu/files/VISTOMS_TUDWingDesign

References

1. 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)
2. Alexandrov, N.M., Lewis, R.M.: Reconfigurability in mdo problem synthesis, part 1. In: Proceedings of the 10th AIAA/ISSMO multidisciplinary analysis and optimization conference, AIAA paper, vol. 4307 (2004)
3. Atkinson, N.: bihisankey.js package—bidirectional hierarchical sankey diagram. https://github.com/Neilos/bihisankey (2015)
4. 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)
5. Bostock, M.: D3.js website. (2015). https://d3js.org/
6. Bowcutt, K.: A perspective on the future of aerospace vehicle design. In: 12th AIAA International Space Planes and Hypersonic Systems and Technologies, Norfolk, VA, AIAA Paper, 2003-6957 (2003)
7. Ciampa, P.D., Nagel, B.: Towards the 3rd generation MDO collaboration environment. In: 30th Congress of the International Council of the Aeronautical Sciences (2016)
8. 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)
9. Gazaix, A., Gallard, F., Gachelin, V., Druot, T., Grihon, S., Ambert, V., Guénot, D., Lafage, R., Vanaret, C., Pauwels, B., et al.: Towards the industrialization of new mdo methodologies and tools for aircraft design. In: 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, p. 3149 (2017)
10. 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)

11. van Gent, I., La Rocca, G., Hoogreef, M.F.M.: CMDOWS: a proposed new standard to store and exchange MDO systems. CEAS Aeronaut J (2017)
12. 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)
13. 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)
14. 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)
15. Grosskopf, A., Weske, M., Decker, G.: the process: business process modeling—using Bpmn. Meghan Kiffer Press (2005)
16. Hoogreef, M.: Advise, formalize and integrate mdo architectures—a methodology and implementation. Ph.D. thesis, TU Delft, Delft University of Technology (2017)
17. La Rocca, G.: Knowledge based engineering techniques to support aircraft design and optimization. Delft University of Technology, TU Delft (2011)
18. Lafage, R.: Xdsmjs package—xdsm diagram generator written in javascript (2016). https://github.com/OneraHub/XDSMjs
19. Lambe, A.B., Martins, J.R.: Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. Struct. Multidiscip. Optim. **46**(2), 273–284 (2012)
20. Lano, R.J.: The n2 chart. TRW Software Series (1977)
21. 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)
22. 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)
23. Pate, D., Gray, J., German, B.: A graph theoretic approach to problem formulation for multidisciplinary design analysis and optimization. Struct. Multidiscip. Optim. **49**(5), 743–760 (2014)
24. Sankey, H.R.: The thermal efficiency of steam engines. Minutes of the Proceedings of the Institution of Civil Engineers pp. 182–212 (1896). https://doi.org/10.1680/imotp.1896.19564
25. Schmidt, M.: Der Einsatz von Sankey-Diagrammen im Stoffstrommanagement. Beiträge der Hochschule Pforzheim. Hochsch. Pforzheim (2006). https://books.google.de/books?id=zhBbMgAACAAJ
26. Sellar, R., Batill, S., Renaud, J.: Response surface based, concurrent subspace optimization for multidisciplinary system design. AIAA paper **714**, 1996 (1996)
27. Shahpar, S.: Challenges to overcome for routine usage of automatic optimisation in the propulsion industry. Aeronaut. J. **115**(1172), 615 (2011)
28. Simpson, T.W., Martins, J.R.: Multidisciplinary design optimization for complex engineered systems: report from a national science foundation workshop. J. Mech. Des. **133**(10), 101,002 (2011)
29. Steward, D.V.: The design structure system: a method for managing the design of complex systems. IEEE Trans. Eng. Manag., **3**, 71–74. IEEE (1981)
30. Vandenbrande, J., Grandine, T., Hogan, T.: The search for the perfect body: Shape control for multidisciplinary design optimization. In: 44th AIAA Aerospace Science Meeting and Exhibit, Reno, NV, 2006-928 (2006)
31. Wagner, T.C., Papalambros, P.Y.: General framework for decomposition analysis in optimal design. ASME Adv. Des. Autom. New York **65**, 315–325 (1993)