A critical look at design automation solutions for collaborative MDO in the AGILE paradigm

van Gent, Imco; Aigner, Benedikt; Beijer, Bastiaan; La Rocca, Gianfranco

DOI
10.2514/6.2018-3251

Publication date
2018

Document Version
Accepted author manuscript

Published in
2018 Multidisciplinary Analysis and Optimization Conference

A Critical Look at Design Automation Solutions for Collaborative MDO in the AGILE Paradigm

Imco van Gent*
Faculty of Aerospace Engineering, TU Delft, Kluyverweg 1, 2629 HS, Delft, The Netherlands

Benedikt Aigner†
Institute of Aerospace Systems, RWTH Aachen University, Wuellnerstrasse 7, 52062, Aachen, Germany

Bastiaan Beijer‡
KE-works, Molengraaffsingel 12, 2629 JD, Delft, The Netherlands

Gianfranco La Rocca§
Faculty of Aerospace Engineering, TU Delft, Kluyverweg 1, 2629 HS, Delft, The Netherlands

This paper presents a critical discussion on the automated problem formulation and workflow creation approach developed within the European project AGILE to support and accelerate the setup of aircraft MDO workflows in a large, heterogeneous team of experts. The developed framework is based on a methodological approach, called the *AGILE paradigm*, where a complete MDO system is formulated and executed in five main steps. In Step I the requirements are collected, in Step II a repository of disciplinary tools is established, in Step III the design optimization problem is formulated and structured according to a selected MDO architecture, in Step IV an executable workflow is assembled and finally operated in Step V. All steps have been streamlined and highly automated through the development of a novel set of MDO support applications and data standards, addressed as the *AGILE MDO framework*. This framework was tested through a series of design campaigns culminating with four design tasks, where a variety of unconventional aircraft configurations is collaboratively designed using MDO. After a brief introduction on the AGILE paradigm and the four design tasks, this paper will focus on a set of AGILE framework core components enabling the automated process to formulate and execute collaborative MDO systems. The strengths and current limitations of these components are discussed, based on the extensive feedback from the heterogeneous set of specialists involved in the four design tasks. Although drastic reductions in the setup time of an MDO system (up to 40 percent) appear to be already achievable, recommendations are provided to improve the flexibility, usability and scalability of the framework.

Nomenclature

AGILE	=	Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts	CFSE	=	Computational Fluids and Structures Engineering
AIRI	=	Airinnova	CGNS	=	CFD General Notation System
API	=	Application Programming Interface	CIAM	=	Central Institute of Aviation Motors
BLI	=	Boundary Layer Ingestion	CMDOWS	=	Common MDO Workflow Schema
BWA	=	Box-Wing Aircraft	COC	=	Cash Operating Cost
BWB	=	Blended-Wing Body	CPACS	=	Common Parametric Aircraft Configuration Schema
CAD	=	Computer-Aided Design	DLR	=	German Aerospace Center

*Ph.D. Student, Flight Performance and Propulsion Section, i.vangent@tudelft.nl, AIAA student member
†Ph.D. Student, Institute of Aerospace Systems, aigner@ilr.rwth-aachen.de
‡Project Engineer, Solutions Department, bastiaan.beijer@ke-works.com
§Assistant Professor, Flight Performance and Propulsion Section, g.larocca@tudelft.nl, AIAA member

DOC	=	Direct operating Cost
DOE	=	Design Of Experiments
DUT	=	Delft University of Technology
GUI	=	Graphical User Interface
IDF	=	Individual Discipline Feasible
I/O	=	Input/Output
KADMOS	=	Knowledge- and graph-based Agile Design for Multidisciplinary Optimization System
KBE	=	Knowledge-Based Engineering
LCC	=	Life Cycle Cost
MDA	=	Multidisciplinary Design Analysis
MDF	=	MultiDisciplinary Feasible
MDO	=	Multidisciplinary Design Optimization
NLR	=	Netherlands Aerospace Centre
ONERA	=	French Aerospace Lab
PIDO	=	Process Integration and Design Optimization
PoliTo	=	Politecnico di Torino
RCE	=	Remote Component Environment
RWTH	=	Rheinisch-Westfälische Technische Hochschule
SBW	=	Strut-Braced Wing
SMR	=	Surrogate Model Repository
STEP	=	Standard for the Exchange of Product model data
TPA	=	TurboProp Aircraft
TsAGI	=	Tsentralniy Aerogidrodinamicheskiy Institut
UNINA	=	Università degli Studi di Napoli Federico II
VISTOMS	=	VISualization TOol for MDO Systems
XML	=	eXtensible Markup Language
XDSM	=	eXtended Design Structure Matrix

I. Introduction

Multidisciplinary Design Optimization (MDO) has been a promising design strategy for at least three decades. However its application, especially in the aeronautic sector, has not been as extensive as expected. Both technical and non-technical challenges are still hampering its widespread exploitation [1–3]. At the highest level, two main phases can be identified for any design and optimization process: the formulation (or setup) phase and the execution phase. Both phases present their specific challenges. For example, the setup phase is affected by the difficulty to select and organize different disciplinary tools, often provided by multiple organizations, into a coherent and consistent process, thereby creating a distributed and collaborative MDO system. During the execution phase, the main challenges are related to the integration of the previously defined MDO system into an executable workflow, while accounting for remote tool accessibility and operation, management and protection of large sets of I/O data, computational infrastructure administration, etc.

Many of the challenges for performing MDO are addressed by AGILE (Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts)*, an EU research project under the funding scheme Horizon 2020 and coordinated by the German Aerospace Center (DLR). AGILE is developing a next generation aircraft MDO framework, which targets significant reductions in aircraft development costs and time to market, leading to cost-effective and greener aircraft solutions. The solutions developed in AGILE are tested throughout a series of design campaigns, including design tasks of increasing complexity. The resulting methods and technical implementations form the so-called *AGILE paradigm*[4] and *MDO framework*[5, 6], respectively. These developments offer a novel design methodology and technical framework aimed at accelerating the deployment of collaborative, large-scale design and optimization systems, in heterogeneous teams of experts.

The AGILE paradigm is based on the five-step development process described in Figure 1. In Step I the requirements for the MDO system are collected, in Step II a repository of disciplinary tools is established, in Step III the design optimization problem is formulated and structured according to a selected MDO architecture, in Step IV an executable workflow is assembled and finally operated in Step V. The first three steps address the formulation phase of the collaborative MDO system, while the last two concerns with the execution phase. As shown in the right column of Figure 1, a number of components (i.e. software applications and data standards) is involved in one or more steps of the overall AGILE paradigm.

In the next section, the five-step approach will be elaborated in more detail, with specific emphasis on Steps II to IV

*`http://www.agile-project.eu`, accessed: May 16th 2018

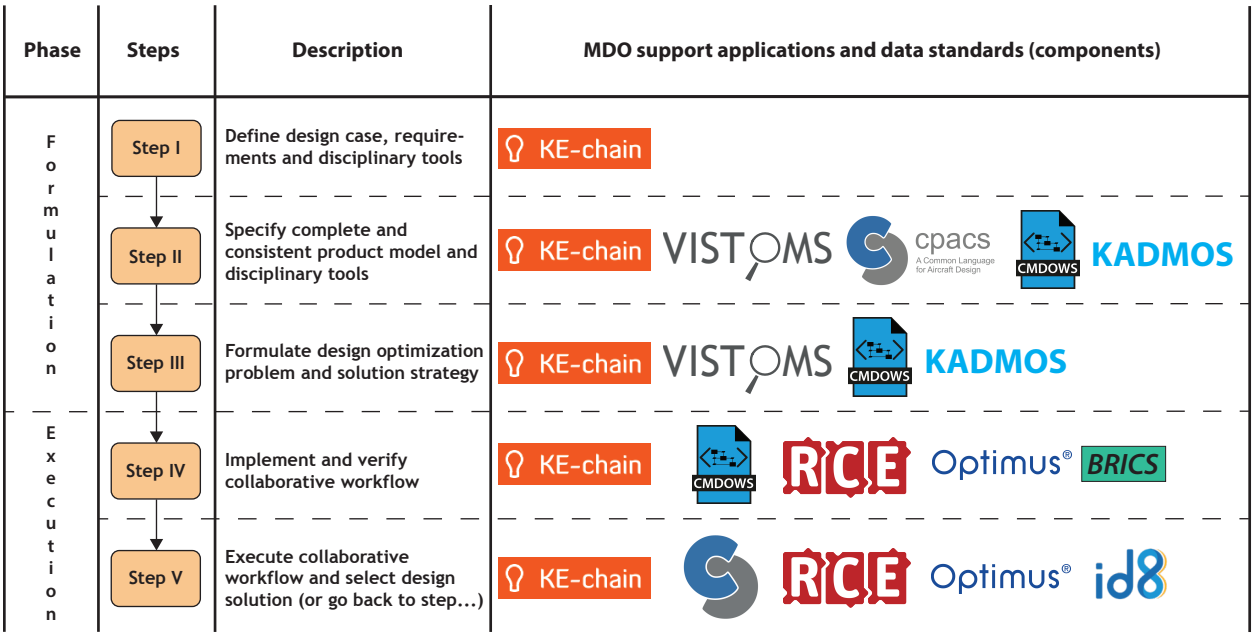

Fig. 1 AGILE five-step development process (adapted from earlier work[5])

and six of the key components that have enabled the automation of the MDO system development process in AGILE. In Section III, four design tasks from the last AGILE design campaign are briefly introduced, where the capabilities of the aforementioned six components have been put to test, namely the KE-chain, KADMOS, VISTOMS and RCE applications, and the CPACS and CMDOWS data standards. Section IV is the core of this paper and provides a critical assessment of the aforementioned six components and the implemented AGILE framework as a whole. On the basis of the extensive feedback from the AGILE partners (including three aircraft manufacturers, three research institutes, four universities and four software solution and engineering service providers) involved in the four design tasks, strengths and current limitations are discussed. An estimation of the achieved gains in terms of time reduction and general support in applying MDO in a collaborative environment is provided, together with a list of recommendations for further development.

II. Automation approach

As introduced above, the MDO system formulation and workflow creation process is formalized through the five-step approach and supported by a collection of applications and data standards that form the AGILE paradigm and the AGILE MDO framework respectively (Figure 1). A more detailed illustration of the automation approach discussed in this paper, with specific focus on Steps II to IV, is provided in Figure 2 and briefly discussed in this section. For a comprehensive description of all steps, support applications and data standards the reader is referred to earlier publications.[4–6]

A. Step I

In this step (outside the focus of this paper) the process integration platform KE-chain[†] is used to initiate the design task (e.g. the MDO of a given aircraft configuration) by storing its requirements and building an inventory of the available disciplinary tools. As shown in Figure 1, KE-chain is actually used in all steps of the AGILE paradigm, playing the role of graphical user interface (GUI) and "control room" for the whole AGILE framework. In most of the other steps, other applications are actually responsible for the automation of the process, while operating "under the hood" via KE-chain's open-source Application Programming Interface (API) Pykechain. Hence, *KE-chain as MDO development process integration platform* is the first component (C1) of the automation approach discussed in this paper.

[†]`http://www.ke-works.com/platform`, accessed: May 16th 2018

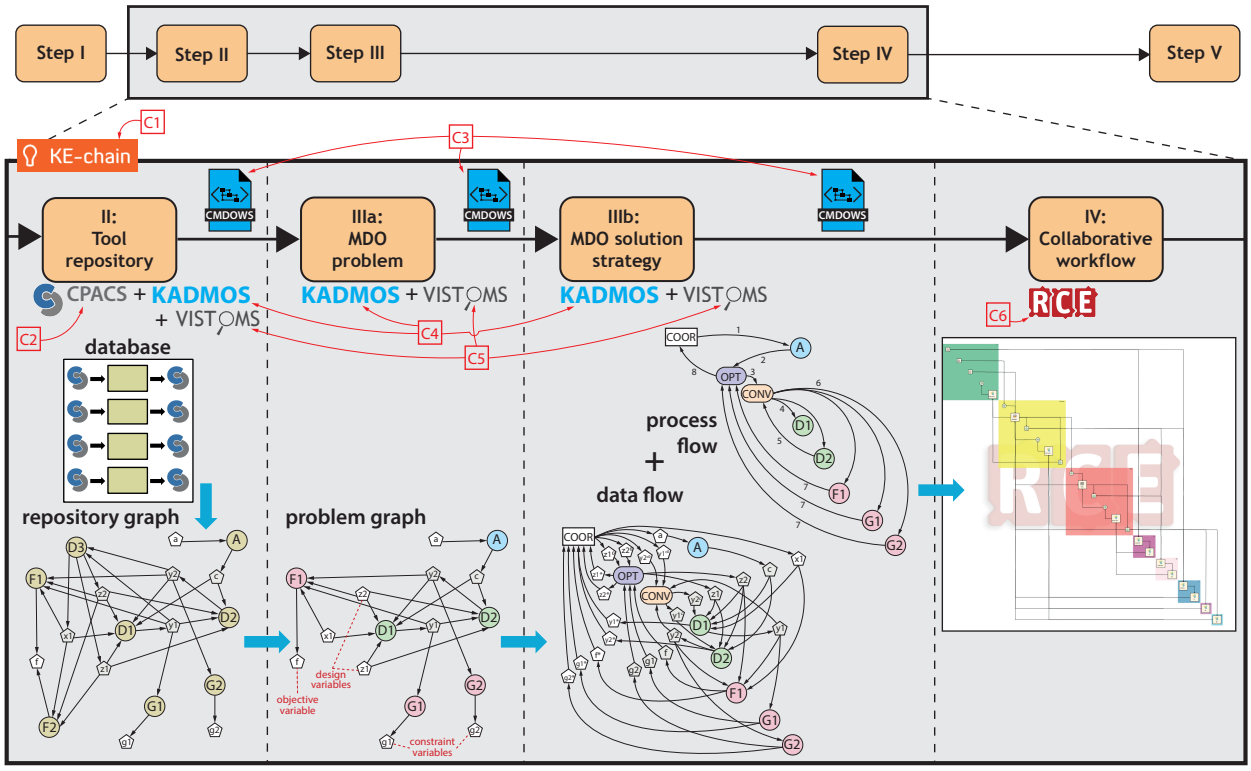

Fig. 2 Detailed overview of the automation approach discussed in this paper (graphs and workflow screenshot based on the implementation of an MDO benchmark problem: Sellar problem [7])

B. Step II

In Step II the creation of the product model and the assembly of the repository of disciplinary tools are taking place. The product model is a structured description (e.g. geometry, analysis results, etc.) of the design that is under consideration. In AGILE, an XML-based central data schema is used as template for this product model, called Common Parametric Aircraft Configuration Schema (CPACS).[8] XML files following this schema (called CPACS files) are used for the exchange of aircraft data between the various disciplinary tools. Hence, each new aircraft design project builds up its own product model using CPACS as a template to, for example, store a tube-and-wing, box-wing, or strut-braced wing aircraft configuration.

CPACS as data standard for tool integration within the AGILE framework is the second component (C2) discussed in this work. CPACS has been under development for more than ten years involving specialists from all major aircraft design disciplines. Together with KE-chain, CPACS is the only component discussed in this paper that precedes AGILE. The CPACS XML schema is available as part of a broader ecosystem including two support libraries: the TIXI file handling library and the TiGL geometry visualization and processing library. All disciplinary tools are expected to read and write CPACS-based files, though for different disciplinary tasks, different parts of the CPACS schema might be used. The disciplinary tools made available by the various discipline specialists are registered in KE-chain by uploading their I/O CPACS files with respect to the product model. The tool repository itself, which consists of the set of tools and their I/O interconnections, is then stored by calling a KADMOS method from KE-chain and using a second standardized file format: Common MDO Workflow Schema (CMDOWS).[9]

CMDOWS, the third component (C3) discussed in this paper, is another XML-based schema developed in AGILE with a twofold objective: 1) To *facilitate the integration of the various MDO support applications* in the AGILE framework, as visualized in Figure 3. 2) To *enable the storage of an MDO system at different stages* of the five-step approach: tool repository, MDO problem, and MDO solution strategy, as visualized in Figure 2.

CMDOWS is based on the idea of storing MDO systems as (directed) graphs. This because originally CMDOWS was developed to store the output generated by the MDO formulation platform KADMOS (Knowledge- and graph-based Agile Design for Multidisciplinary Optimization System)[10], which is the actual automation engine for the whole

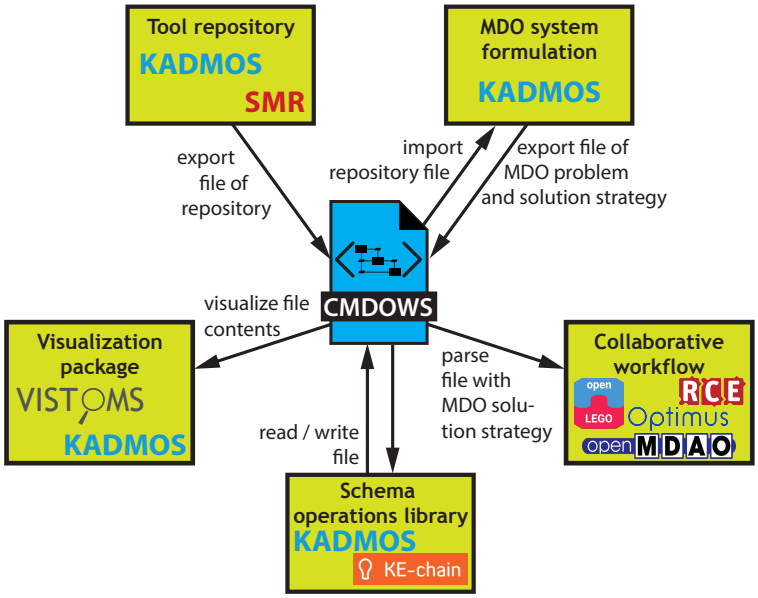

Fig. 3 The CMDOWS format as central data schema to link different MDO support applications including the applications used within AGILE (adapted from earlier work[9])

MDO system formulation process in the AGILE framework. KADMOS starts operating in Step II, where, as illustrated in Figure 2, it imports the tool repository definition stored in KE-chain and transforms it in a directed graph that is compliant to strict graph-theoretic conditions.[10] This will enable KADMOS to run its graph diagnostic methods on the tools' I/O data interconnections, checking for eventual inconsistencies and conflicts. Then the tool repository graph, stored as CMDOWS file, is ready for further manipulations in Step III.

C. Step III

In Step IIIa, the fundamental design optimization (or analysis) problem is formulated. The design team (via KE-chain) selects the required disciplinary tools to solve the problem at hand and marks the elements from the CPACS-based product model according to their role in the design problem, i.e. design variables, objectives, quantities of interest and constraints. On the basis of this input, KADMOS transforms the previously defined tool repository graph into the so-called problem graph (Figure 2), which is again stored as a CMDOWS file for further manipulation.

In the Step IIIB, KADMOS graph manipulation algorithms are again in action to automatically transform the design optimization problem graph according to the MDO solution strategy selected by the design team (again via KE-chain). These solution strategies usually follow one of the available analysis and optimization architectures[11] available in KADMOS, such as multidisciplinary design analysis (MDA), design of experiments (DOE), multidisciplinary feasible (MDF), and individual discipline feasible (IDF). Directed graphs are used again to represent both the data and the process flow required to execute the solution strategy (Figure 2). *KADMOS as platform to formulate MDO systems from tool repository to solution strategy* is the fourth component (C4) discussed in this paper. This solution strategy is then stored in a CMDOWS file (making now use of the full extent of the schema), ready for Step IV.

While the graph representations manipulated by KADMOS allow to fully automate the whole formulation process (from tool repository to fully formalized computational system according to a selected MDO architecture), virtually without any problem size limit, they are not really suitable for human readability and inspection, even in their CMDOWS representation. This challenge is tackled by *VISTOMS (VISualization TOol for MDO Systems)*[12], *the visualization tool for the design team* and the fifth key component (C5) of the MDO framework addressed in this paper. VISTOMS is another of the applications fully developed within AGILE, that reads and processes CMDOWS files to enable discipline specialists, MDO architects and integrators to inspect and debug the MDO system under development by means of comprehensive but human-readable visualizations, such as eXtended Design Structure Matrices (XDSM)[13] and data trees.

VISTOMS is a web-based application built on top of the open source D3.js library and other packages[‡§¶] that enable the dynamic display of data. With conventional visualizations, especially in the case of large computational systems featuring multiple disciplinary tools and a sheer amount of data connections, it is almost impossible to render the full system description without paying either in terms of completeness or clarity. On the contrary, VISTOMS allows inspecting details of MDO system of virtually any size, such as specific information on any system tool and every single piece of exchanged data. It also leverages on KADMOS methods to detect and highlight missing tool inputs and output collisions. These inspections can be done on demand via right-click mouse operations and mouse hovering, and are supported by the zooming and panning operations typical of digital visualizations. Since all VISTOMS visualizations are based on the data stored in CMDOWS files, the design team can inspect the MDO system throughout all the development steps, starting from the initial definition of the tool repository, up to the fully formalized MDO solution strategy.

D. Step IV

The outcome of Step III is the CMDOWS file produced by KADMOS containing the complete formulation of the MDO system. However, such a CMDOWS file is just a formal representation of the MDO system that is not yet executable. The executable workflow matching the solution strategy will have to be build in a specific workflow software. In AGILE, two such Process Integration and Design Optimization (PIDO) software are provided by consortium partners: RCE and Optimus.[14, 15] In this paper the focus is on RCE workflows, though the same workflows can be built in Optimus (or any other workflow software that is able to parse the CMDOWS file for that matter). Hence, the sixth and final component (C6) of the automation approach in the AGILE paradigm is *RCE as CMDOWS parser to create executable workflows*.

The parsing of the CMDOWS file is the automated way to build the collaborative workflow in Step IV. The collaborative workflows which are created in this step are distributed, heterogeneous workflows containing disciplinary tools from different partners that are geographically distributed. The integration of workflow elements that run on different software domains is enabled by the Brics software.[16] Brics is not discussed in this paper, but the interested reader is referred to other work.[6]

E. Step V

The creation of the executable workflow achieved in Step IV marks the end of the automation process discussed in this paper, thus Step V is outside the scope. However, it is important to remark that the computation results obtained in this phase can be used to trigger a new iteration of the previous automation process. For example, the results from a DOE study could be used to go back to Step III to formulate a new MDO problem with different design variables and constraints, or even back to Step II to add a new disciplinary tools as required to increase the level of fidelity or the number of disciplines in the MDO system. While in the traditional way of performing MDO these iterations would imply major and laborious reworks of the existing MDO system, the automation process provided by the AGILE framework can dramatically reduce the overhead associated with each iteration.

III. Design tasks: four unconventional aircraft design cases to test the AGILE framework

In the final year of the AGILE project, a series of diverse aircraft design cases (addressed in the project as *design tasks*) is being performed, with the main goal of testing the AGILE framework and, in particular, to evaluate the performance of the automation process described in the previous section and its six key components:

C1: KE-chain as MDO development process integration platform
C2: CPACS as data standard for tool integration
C3: CMDOWS as data standard to integrate MDO support applications and store MDO systems
C4: VISTOMS as visualization tool for the design team
C5: KADMOS as platform to formulate MDO systems from tool repository to solution strategy
C6: RCE as CMDOWS parser to create executable workflows

In this work, only the four AGILE design tasks relative to unconventional aircraft configurations are considered, as the most challenging experiments to test the developed automation approach (Figure 2). These tasks are briefly introduced

[‡]D3.js website by M. Bostock: `https://d3js.org/`, accessed: May 16th 2018
[§]bihisankey.js package by N. Atkinson: `https://github.com/Neilos/bihisankey`, accessed: May 16th 2018
[¶]XDSMjs package by R. Lafage: `https://github.com/OneraHub/XDSMjs`, accessed: May 16th 2018

in the next subsection A. Subsection B will illustrate the current results from the application of the automation approach to formulate MDO solution strategies in the different design tasks.

A. Description

1. Strut-braced wing (SBW)

Involved partners DLR (task leader), CFSE, CIAM, DUT, NLR, PoliTo, RWTH Aachen, TsAGI, UNINA

Design task objective The emphasis in this task is on the integration of the wing-strut system and the management of the loads on the large span wing. The combination of structural and aerodynamic analysis is of key importance to this task. To start, the operational requirements will be kept constant while the geometrical parameters of the wing-strut system (e.g. aspect ratio, span wise position of the wing-strut connection and struts' thickness-to-chord ratio) will be set as design variables for a DOE study. Then, also changes on the aircraft operational specifications (e.g. cruise Mach number and altitude) will be investigated. The strut-braced wing configuration will be compared to the conventional aircraft designed in earlier AGILE design campaigns and optimized to minimize Direct Operating Cost (DOC), Cash Operating Cost (COC), Life Cycle Cost (LCC), mission fuel, and/or other composite functions.

Automation challenges A large and heterogeneous set of design and analysis tools (e.g. flight dynamics, on-board systems, aerodynamics, structures, costs) from many different partners must be integrated in different collaborative workflows. Severe reconfigurations are envisioned to move from the DOE studies based on configuration parameters, to the design studies with varying operational requirements, and finally to the setup of an optimization process to minimize one of the cost metrics. As the design task develops, the use of higher fidelity tools is envisioned.

2. Box-wing aircraft (BWA)

Involved partners ONERA (task leader), AIRI, CIAM, DLR, DUT, PoliTo, UNINA, RWTH Aachen

Design task objective The design scope for this configuration includes both mission parameters and aircraft design parameters. The objective is to first identify the best BWA configurations for a wide range of missions parameters, then to select the type of mission giving the BWA superiority with respect to conventional aircraft designs. The main objective of the optimization will be minimization of DOC. The broad multidisciplinary focus for this configuration includes the aerodynamic characterization of the box-wing architecture, including control surface behaviour and high-lift capabilities; the structural design of both wings and fuselage including the aeroelastic effect on the wing sizing; more electric on-board system architecture (evaluation of system impact on the overall aircraft design, fuel trim system); stability and control and flying qualities including the definition of a dedicated flight control system.

Automation challenges An extensive heterogeneous set of design and analysis tools from many different partners must be integrated in a design workflow, with a step-wise approach. Different levels of fidelity will be required, ranging from fast semi-empirical methods up to high-fidelity CFD- and FEM-based analysis to assess the aeroelastic behaviour of the airframe and the performance of the novel system of control surfaces distributed over the boxed wing.

3. Blended-wing body (BWB) novel propulsion/airframe integration

Involved partners CFSE (task leader), CIAM, DLR, DUT, PoliTo, RWTH Aachen, TsAGI, UNINA

Design task objective The objective for this novel configuration is to minimize the fuel consumption for a maximum range constraint at maximum payload, focusing mainly on the embedded engines integration aspects, based on Boundary Layer Ingestion (BLI). The definition of the number and location of the engines, the design of the air intakes for active BLI control and the aerodynamic characterization of the full configuration are the main aspects of this optimization study.

Automation challenges Heterogeneous set of high-fidelity design and analysis tools (e.g. aerodynamics, on-board systems for BLI control, engine integration) from different partners should be integrated in a collaborative workflow. Due to the use of tools that are not fully CPACS-compatible (mainly the high-fidelity aerodynamic tools requiring accurate CAD files as input), the automated integration and execution of the workflow is expected to be particularly challenging.

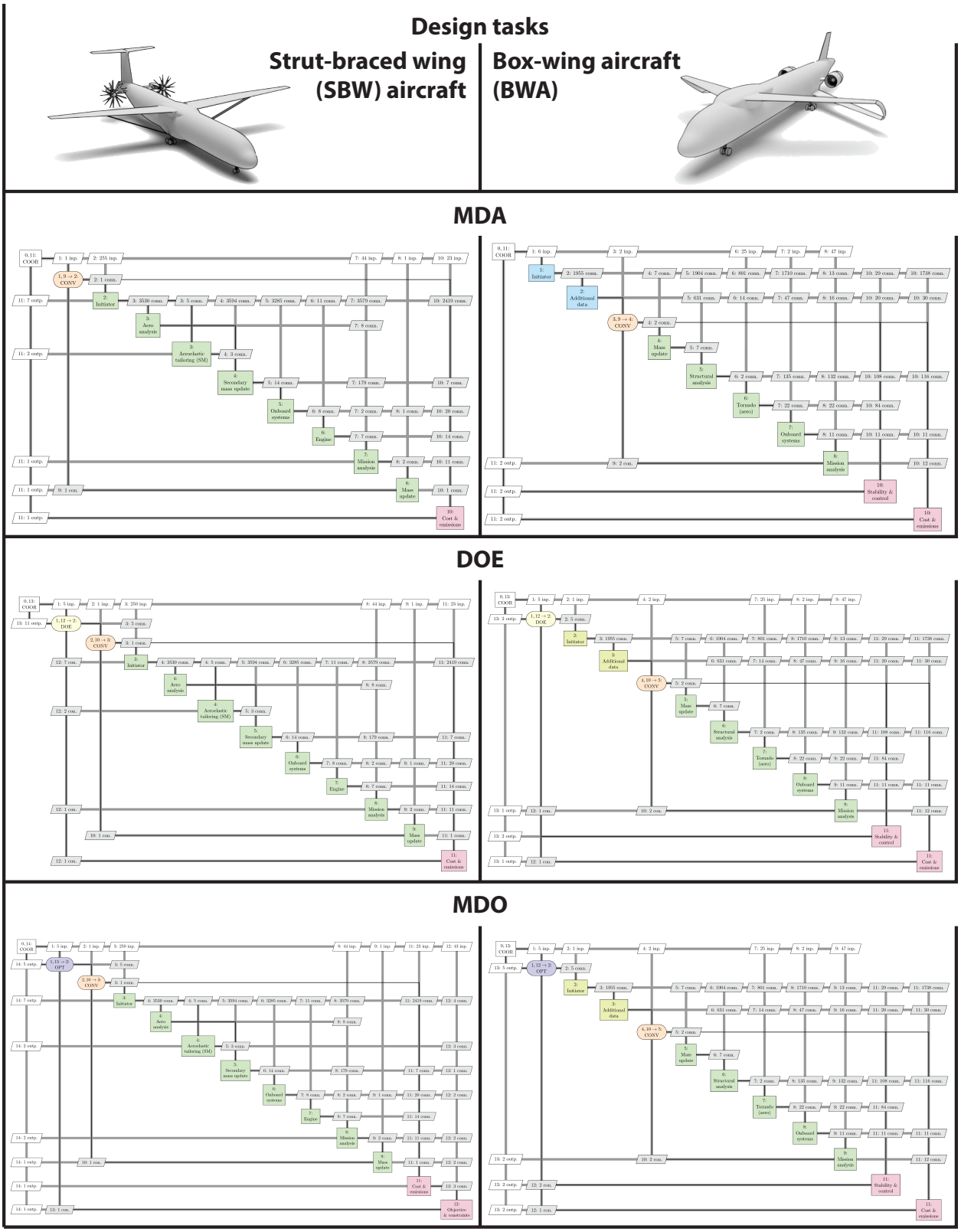

Fig. 4 Snapshots of the XDSMs of the formulated workflows for the SBW and BWA design tasks. Each XDSM corresponds to a CMDOWS file (C3) containing the neutral definition of the MDO solution strategy.

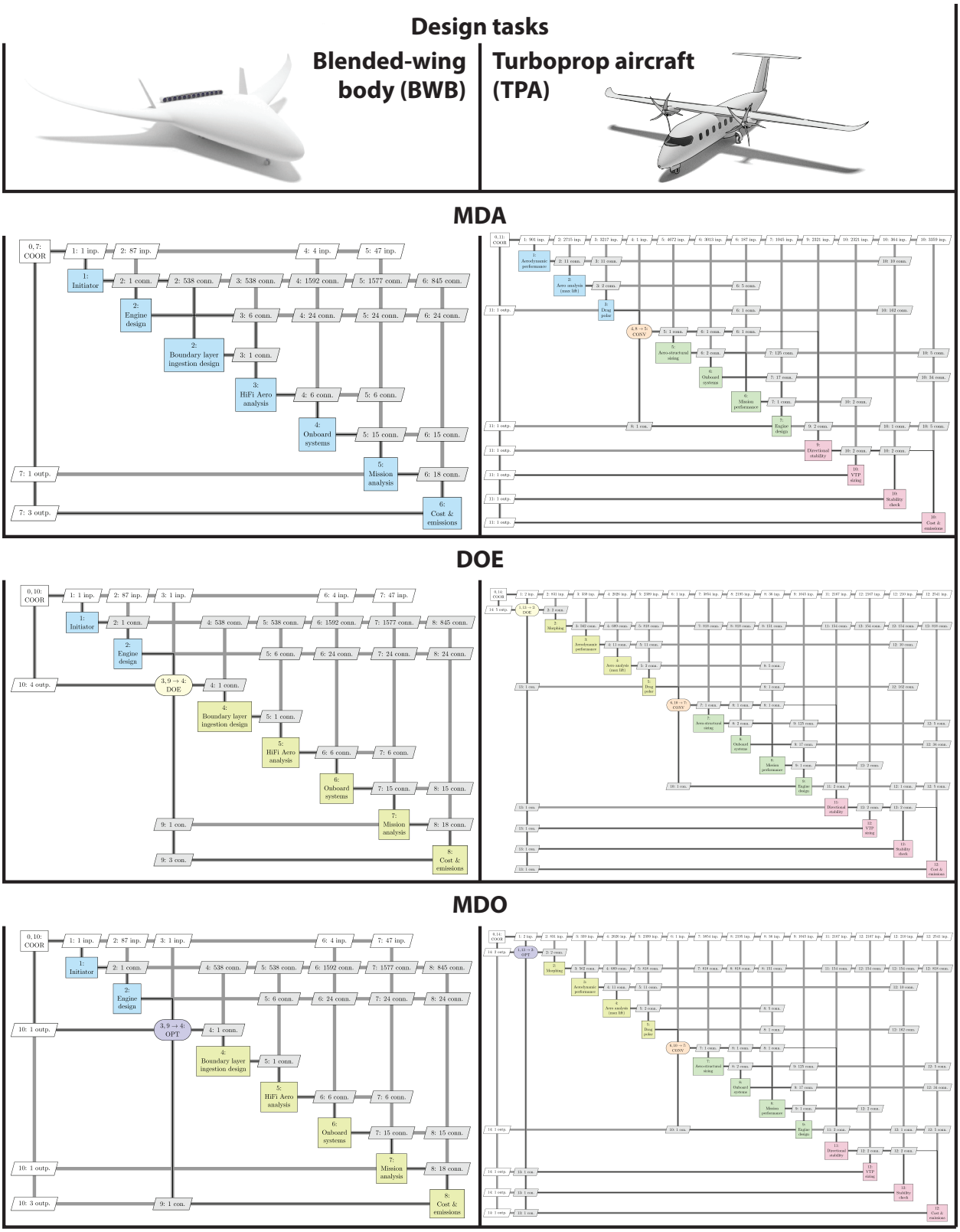

Fig. 5 Snapshots of the XDSMs of the formulated workflows for the BWB and TPA design tasks. Each XDSM corresponds to a CMDOWS file (C3) containing the neutral definition of the MDO solution strategy.

4. Turboprop aircraft (TPA)

Involved partners UNINA (task leader), Airbus D&S, AIRI, CFSE, CIAM, DLR, DUT, PoliTo, RWTH Aachen, UNINA,

Design task objective This design task aims at comparing a novel turboprop configuration with fuselage-mounted engines with a more conventional wing-podded version. The main focus of the task concerns the minimization of DOC, while including noise constraints in accordance with regulations. Main design variables will concern the geometry of the wing for the two configurations: wing area, wing span, wing thickness ratio and wing taper ratio. A key constraint is the wing sweep angle which must be kept minimum in view of exploiting the drag reduction of natural laminar flow wings.

Automation challenges As for the other design tasks, the integration of a heterogeneous set of design and analysis tools from many different partners makes for the main challenge. In this case, the need to execute two optimization studies on two aircraft configurations will put extra constraints on the overall computational performance.

B. Automation process application in the design tasks

At the submission time of this paper, all the design teams working on the four design tasks have used the AGILE framework to formulated workflows of increasing complexity, based on their own task objectives. A top-level overview of the results generated by the automated process discussed above is summarized in Figures 4 and 5. Each XDSM in these figures represent a VISTOMS (C5) visualization of a CMDOWS (C3) file generated by KADMOS (C5), containing the formulation of a certain MDO solution strategy. Finally, the CMDOWS files were parsed in RCE (C6) to obtain executable workflows. One of the automatically generated RCE workflows, relative to the BWA design task is shown later in this paper.

First, all design tasks configured a simple MDA to test the different tools and converge a single design point, see second row in Figures 4 and 5. Subsequently, DOEs were configured to explore the design space, using the previously tested MDA, third row in the snapshot figures. In some cases, DOEs have been set up in view of generating surrogate models for use in later optimization runs. Finally, MDO architectures, such as MDF and IDF, were formulated to run actual optimizations, the XDSM snapshots are depicted in the last row of Figures 4 and 5.

Here below, the box-wing aircraft (BWA) design task is explained in more detail to provide more insight into the application of the automation approach summarized in Figure 2. A detailed description of the aforementioned design tasks is out of scope of this work, but the interested reader is encouraged to look for future publications from the AGILE consortium.

Step II: To start building the tool repository with CPACS-compatible tools, the product model of a first box-wing aircraft configuration is initialized by the design team. In KE-chain, see Figure 6, this CPACS file can be inspected using the VISTOMS tree viewer and the geometry visualized using a CAD export from the TiGL library. Subsequently, the KE-chain interface is used to upload separate I/O files for each disciplinary tool, as illustrated in Figure 7. The execution button at the bottom of the screenshot triggers a KADMOS method on the KE-chain server to build the tool repository graph. This graph can then be saved as a CMDOWS file to proceed to the next step, as well as for visual inspection with VISTOMS.

Step IIIa: In Step IIIa KE-chain forms are used to specify the input required for KADMOS to build the problem graph, i.e. the definition of the problem roles, such as design variables, constraints, objectives, etc. After processing these forms the MDO problem graph is built and stored as a CMDOWS file, such to proceed to Step IIIb and inspection in VISTOMS, see Figure 8. In the visualization in Figure 8 the elements from CPACS with a special role in the MDO problem (e.g. design variables, objective, constraint) are inspected using the tree viewer.

Step IIIb: After specification of the problem graph, KADMOS automatically imposes the MDO architecture selected by the design team through the KE-chain interface. In this case, the MDF architecture with a Gauss-Seidel convergence scheme was selected. The resulting formulation is again stored as CMDOWS for proceed to the next step or for visual inspection in VISTOMS. The XDSM visualization of the complete MDO system formalization is shown in Figure 9 (this is the same figure as the BWA-MDO snapshot in Figure 4).

Step IV: Finally the CMDOWS file containing the solution strategy formulated by KADMOS is parsed in RCE for the automatic generation of the executable workflow shown in Figure 10.

In the other design tasks the same steps were carried out for the creation of the XDSMs in Figures 4 and 5.

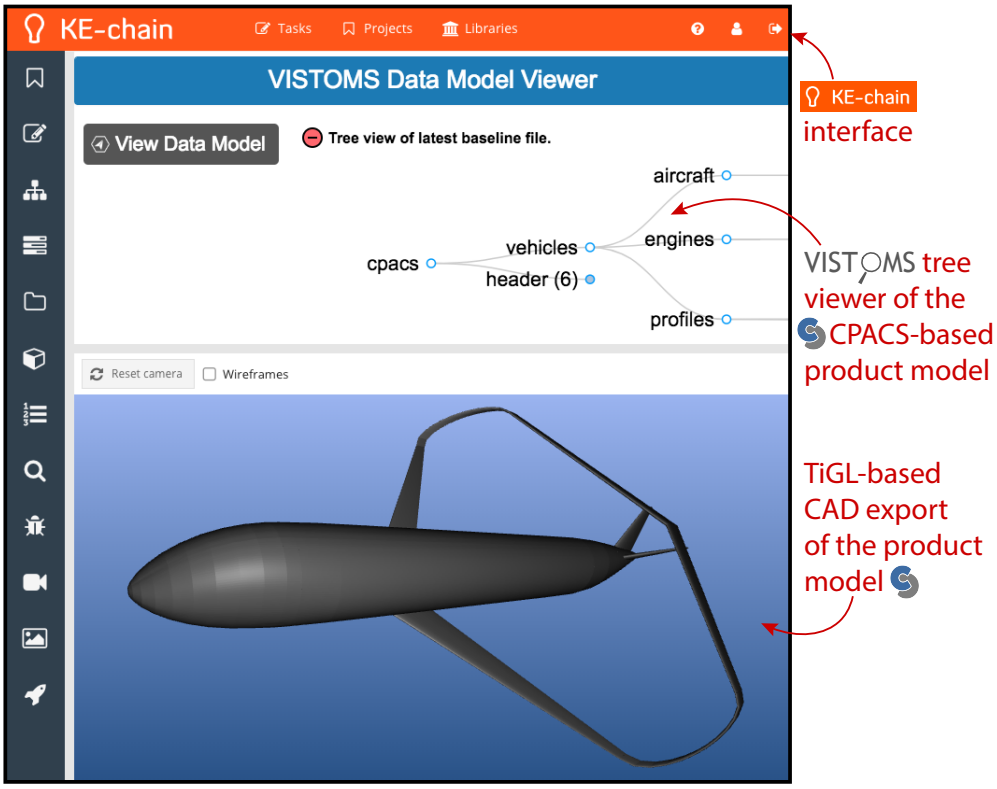

Fig. 6 Screenshot of the KE-chain interface (C1) embedding the VISTOMS (C5) data model viewer and the graphical viewport. The first three levels of the CPACS (C2) product data model used in the BWA design task and the corresponding CAD rendering produced by TiGL are displayed.

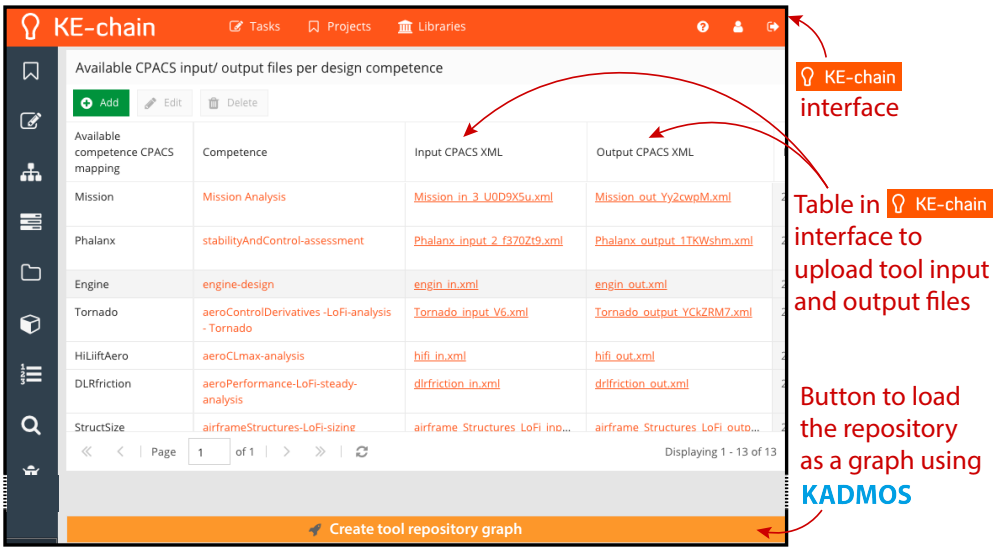

Fig. 7 Step II: Screenshot of the KE-chain interface (C1) depicting the form containing the table to upload the different CPACS input and output files for each tool to build the tool repository.

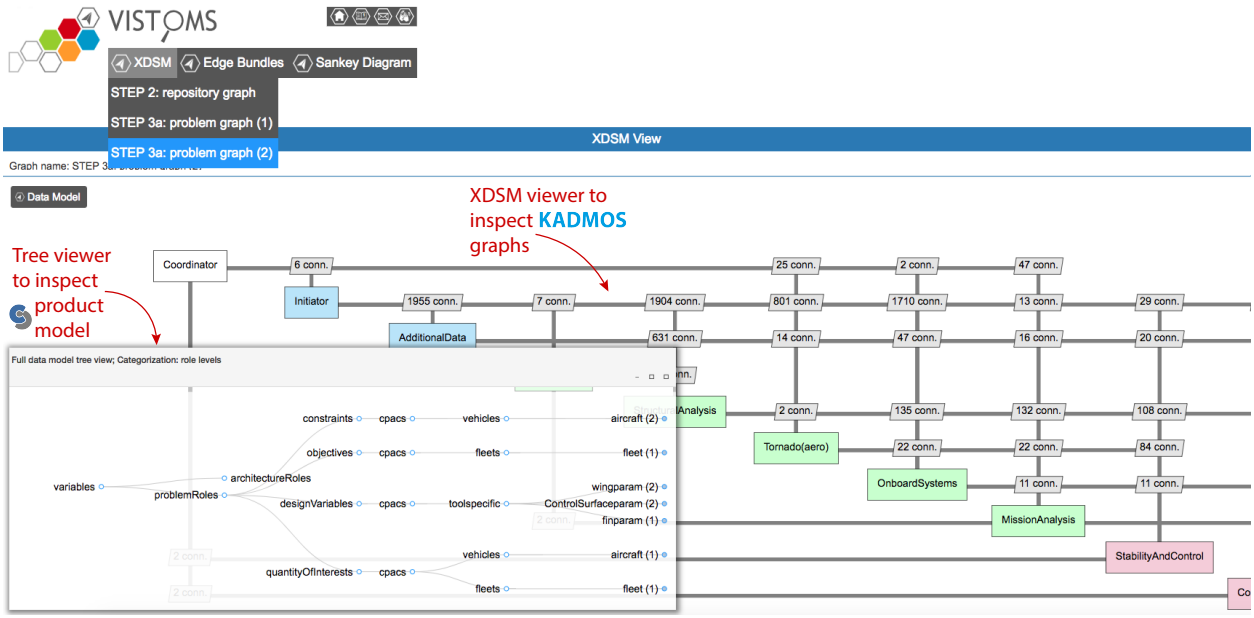

Fig. 8 Step IIIa: Screenshot of the web-based VISTOMS interface (C5) showing an XDSM visualization of the problem graph assembled by KADMOS (C4) and a tree view with some elements from the CPACS (C2) product model, with assigned problem roles.

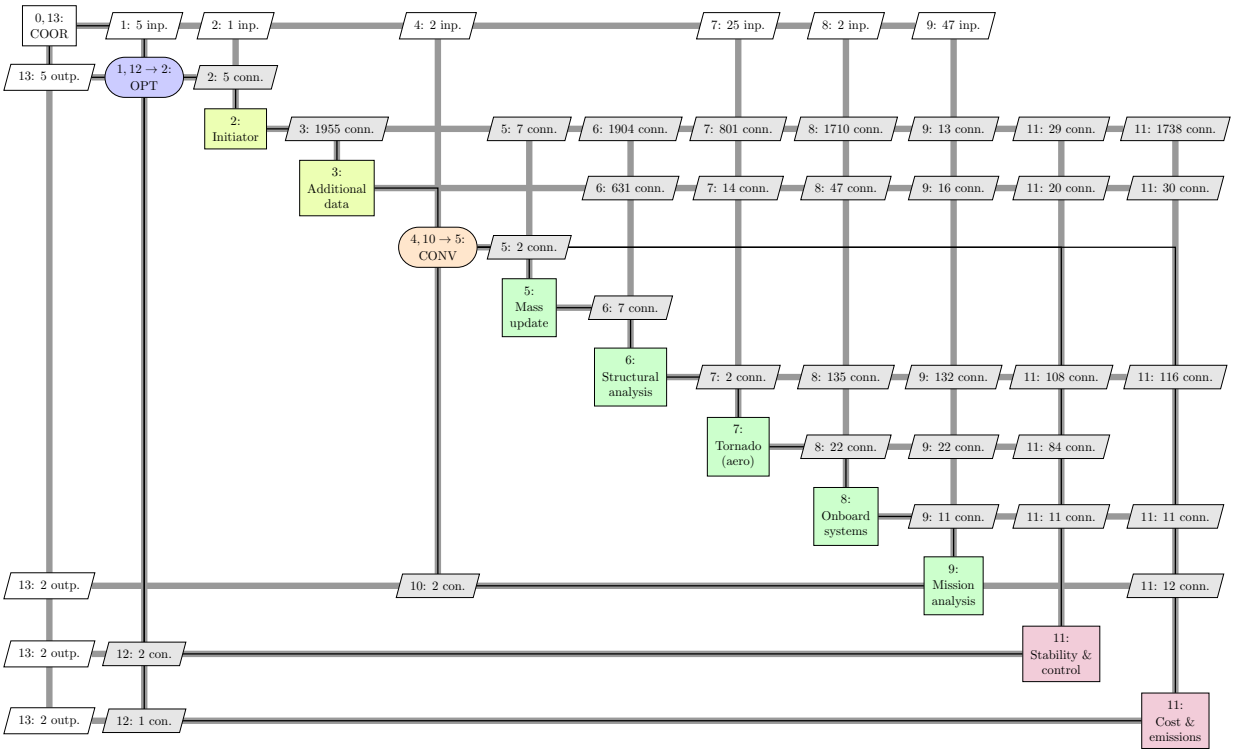

Fig. 9 Step IIIb: XDSM showing the MDO solution strategy formulated by KADMOS (C4) based on the MDF architecture. Same figure as BWA MDO snapshot in Figure 4.

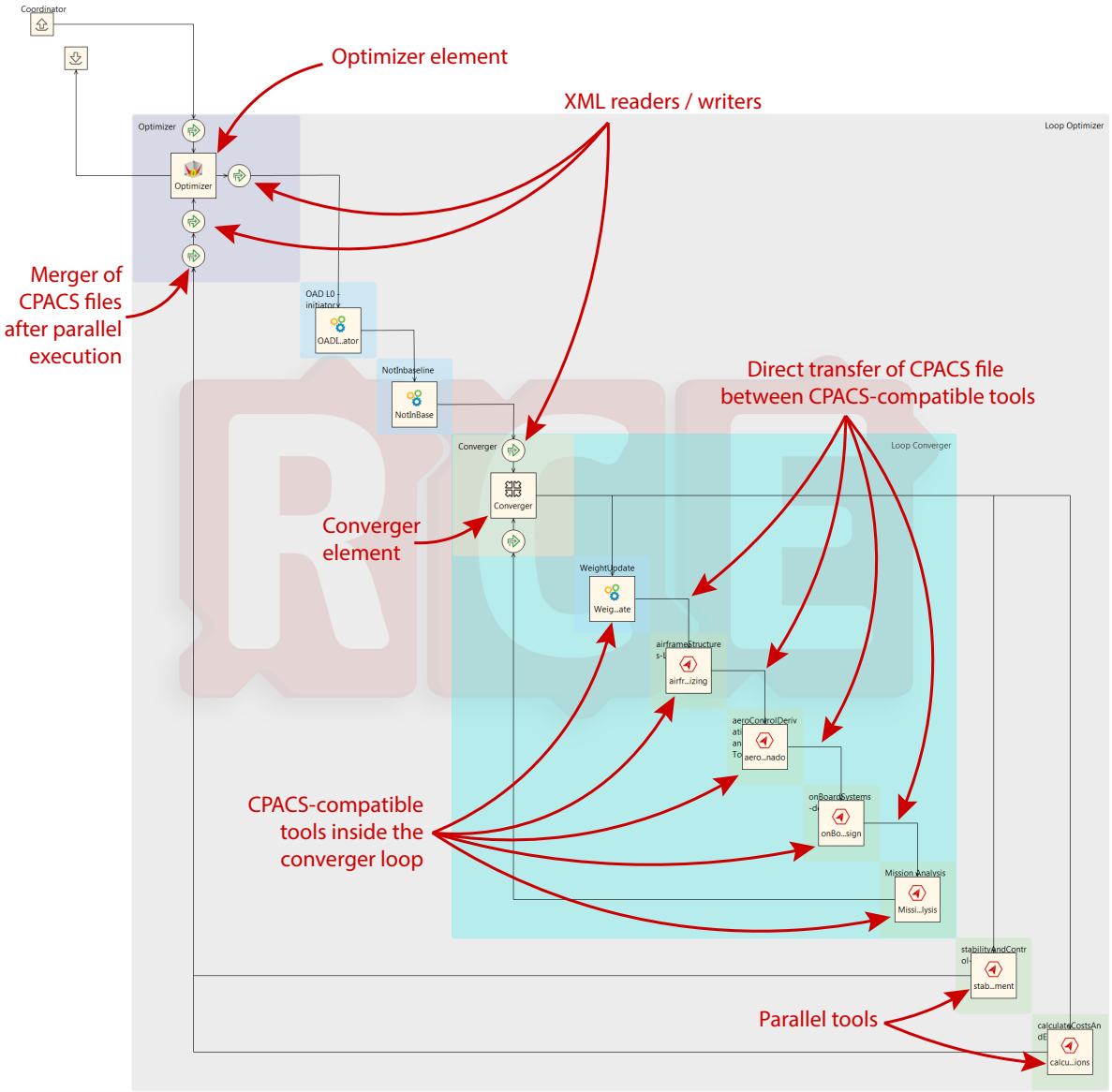

Fig. 10 Step IV: Screenshot of the workflow automatically generated in RCE, by parsing the workflow formulation visualized in Figure 9.

IV. Preliminary assessment of the AGILE automation process

After the formulation of the different MDO workflows, a preliminary feedback of the AGILE consortium partners on the six components of the automation approach were gathered using an extensive survey. In this survey, users were asked to share their experience using the AGILE automation approach and to indicate the strengths, limitations, opportunities and risks for each of the six components, as well as for the AGILE framework as a whole. About thirty AGILE partners have provided their feedback. These partners played different roles within the design tasks, had different levels of experience in using the framework, and worked from different perspectives being based at one of the three aircraft manufacturers, three research institutes, four universities or four software solution and engineering service providers involved in AGILE. The results of the design tasks themselves in combination with the feedback from the survey are elaborately discussed in the next subsections.

A. KE-chain as MDO development process integration platform (C1)

KE-chain successfully enabled heterogeneous teams of experts to formulate, integrate and execute a diverse range of MDO systems. According to the survey, KE-chain was acknowledged to play an important role in the AGILE framework, providing a web-based, collaborative environment (see Figures 6 and 7) able to manage data of multiple parallel design tasks. KE-chain was functioning as the framework cockpit or a kind of control room for multiple users to work together according to a well-structured development process. The success scored by KE-chain as integration platform was mainly due to its flexible GUI, database and the API Pykechain, which enabled the easy integration of applications such as KADMOS, VISTOMS and CPACS viewers on a server.

As KE-chain represented to users the look and feel for the whole MDO framework, it was felt that a balance must be found in hiding complexity by minimizing the number of steps and effort needed to set up an executable MDO workflow, whilst offering the controls to steer the process even for very specific type of problems. One of the main observations from the survey was that the interface in KE-chain has been developed with too much focus on expert users who require full access and control over the MDO system, such as system integrators and architects. For novice users and disciplinary tool specialists that only have to access and control a small part of the MDO system, the framework was found to be too complex. Current documentation and tutorial material was too limited for these users.

Another issue found, was that KE-chain experienced difficulties in handling large scale MDO problems: the web-based interface occasionally slowed down as large amount of data had to be written from and to the database. Related to this, running operations cannot be cancelled or reverted once started on the server. As a consequence, users occasionally have to wait minutes or hours before they can restart an operation.

A limitation encountered by users involved in multiple design tasks was that project-to-project transfer of information is currently not supported by KE-chain. Hence, some manual labor is required to transfer duplicate information (e.g. the I/O CPACS files of a tool used in multiple design tasks) between different design tasks. This is a bottleneck of the current KE-chain version, but it touches upon the fundamental issue of project-to-project information sharing that is further discussed in Section IV.D.

The following priorities are advised for future developments:

- Introduce a user access management system to expose users only to functions and information relevant to their role in the design task. For example, disciplinary specialists, who focus on integrating their tools in the MDO system, should only be allowed to see and use that particular part of the framework.
- Support the termination of scripts and improve the performance of reading and writing information to the database.
- Create extensive documentation and tutorials.
- Provide the ability to exchange and transfer information between projects / design tasks.

To conclude, KE-chain[‖] as an integration platform to enable the AGILE paradigm has proven to be a valuable component. The survey indicated that a time gain of approximately 20% could be achieved by using KE-chain as MDO development process integration platform. However, there was a large difference between the time gains estimated by novice and experienced users. The latter (which include also the AGILE architects and integrators) rate the platform contribution to the time gain significantly higher (up to 80%) compared to new users (and "traditional" design specialists). Some of these stated to give up working on the platform after it did not prove to have added value for them. From one side, it is natural that for a disciplinary expert playing just the role of tool provider, working on the integration platform (and adjusting its tool to make it suitable for the integration) is experienced as pure overhead. A different feedback is

[‖] N.B. In AGILE, KE-chain has been used as an R&D platform. The R&D version is not on a par with the latest version of the commercial platform, as custom developments were required to adjust KE-chain to specific AGILE requirements. An upgrade of the entire AGILE MDO framework to the latest version of KE-chain would already solve some of the limitations mentioned in this section (e.g. script termination).

expected in the moment the AGILE framework will be fully operative, and all tools will be easily re-usable in many different workflows without any extra work required from their side.

B. CPACS as data standard for tool integration (C2)

CPACS is the most mature component in the AGILE MDO framework and enables the integration of a large variety of disciplinary tools from different experts. Being a core component in the AGILE framework (and in the paradigm itself, which is based on the common data model integration approach) all partners had direct experience with it, therefore extensive feedback was received in the survey. The CPACS's role as a common language for collaborative aircraft design was much appreciated. However, the developments in AGILE have reached the boundaries of what CPACS can offer and highlighted some limitations, which will drive the future development agenda of the standard. The main limitations are discussed below.

1. Single representation of all data too inflexible

A key assumption in CPACS is that all data have to be stored in a non-redundant fashion to avoid inconsistencies. Therefore, there is only one valid way to store, for example, certain analysis results or the definition of a wing planform or structural components. This was found to be too limiting in some of the AGILE design tasks. Partners in the BWA and BWB design tasks found the way in which aerodynamic and flight dynamics data are stored not intuitive and impractical. As a consequence, the involved discipline experts needed to translate the data back to a more convenient format. Similarly, another partner found the way of defining the rib position in a wing inadequate, as the schema allows indicating only the number of ribs per segment, but not the specific positioning and the individual properties of those ribs.

In other cases, the geometrical definition required by CPACS appeared be too detailed with respect to the limited set of design variables used by many conceptual tools. For example, a wing in CPACS has to be defined using airfoils, sections, and segments, whereas many conceptual design tools define a wing geometry only by means of high-level parameters, such are aspect ratio, wing span and quarter-chord sweep. Some of these parameters can be determined using the TiGL library, but others need to be derived by the tool provider, by pre-processing the input CPACS file, with the risk of inconsistent interpretation of such key variables among different tools.

2. Missing functionality in in the CPACS ecosystem

Related to the previous issue, it is even more challenging to adjust a geometry stored in CPACS based on high-level variables (e.g. span, aspect ratio). Thus, CPACS post-processing is even more challenging than the pre-processing addressed above, yet the capability to "morph" the product geometry is crucial to set up DOE and MDO workflows. Currently, morphing methods are not yet available in the CPACS ecosystem.[17]. This is related to the lack of a parametric aircraft model linked to CPACS files. The TiGL tool can provide CAD models of the geometry, but it is focused on reading a CPACS file, not changing it. A way forward would be to set up a Knowledge-based Engineering (KBE) tool[18] to handle and adjust CPACS files, which would ultimately act as an interactive CPACS visualizer and editor. Another missed capability, was a set of methods in the TIXI library to automatically track the I/Os of a tool with respect to CPACS. Such methods come in handy for I/O definitions that need to be provided in KE-chain as shown in Figure 7. Alternatively, a KBE-based system could also provide this information using its native dependency tracking algorithm.

3. Inadequate support for high-fidelity analysis tools

The third CPACS limitation encountered in AGILE is related to the support for high-fidelity analyses. In the BWB design task, multiple high-fidelity tools needed to be integrated to analyze the system. However, it was not possible to store the fine details of the geometry that was required for the analyses to run. Therefore a temporary solution was devised based on the exchange of these data using CAD files. First of all, the exchange of files is not formally defined in CPACS, but more importantly, this issue raises the question up to what level an XML-based schema like CPACS can actually still be applied effectively for performing MDO with high-fidelity analysis tools. Typical high-fidelity input (CAD quality geometry) and output (large-scale element/node-based results) data are difficult to store all in one file and the current schema definition in CPACS does not scale well in such cases. Generally, high-fidelity data are stored using dedicated standards (e.g. STEP for CAD geometry, CGNS for CFD data, etc.). It is suggested to add an element definition in CPACS to include such high-fidelity data formats.

4. Improvements in the standard development process

The final CPACS limitation experienced in AGILE is related to the development process of the standard. Many respondents in the survey felt that the development of the schema is too slow, while, at the same time, it was difficult for developers to find the right balance between sticking to the official development process including reviews of CPACS adjustments/extensions and providing ad-hoc "quick and dirty" solutions to project partners. The CPACS development process needs to be professionalized and the release cycle shortened. One respondent suggested to organize an international review committee, similar to the one set up for the CGNS format. The organization of such a committee would also benefit the wider, international adoption of CPACS as an official data standard for conceptual and preliminary aircraft design, such as, for example, the BIM standard in the infrastructure domain.

C. CMDOWS as data standard to integrate MDO support applications and store MDO systems (C3)

CMDOWS has thoroughly demonstrated its capability to act as a data standard to integrate the heterogeneous collection of MDO support applications developed and deployed in AGILE, such as KE-chain, KADMOS and VISTOMS. Within the four design tasks, no major issues or limitations were encountered concerning this component. A minor issue found, was the size of the CMDOWS files, which increase significantly for design tasks involving large MDO systems (>100MB for a single XML file) and a lot of CPACS elements (>4000).

Proof of the increased agility offered by the adoption of CMDOWS was provided when, in the later development stage of AGILE, new support applications needed to be integrated in the framework, such as the Surrogate Model Repository (SMR) and OpenMDAO (Figure 3). The integration of the SMR in the framework involved links to KE-chain, KADMOS and VISTOMS. This was achieved in a short time using CMDOWS, without requiring significant changes or adjustments of any of the involved applications. Similarly, the OpenMDAO platform was conveniently linked to the broader framework through a newly developed Python package called OpenLEGO[19].

Limitations of the schema mainly stem from its relatively low level of maturity. Contrarily to CPACS, CMDOWS was developed from scratch within the AGILE project and has therefore only been tested and applied using AGILE design tasks. Therefore, new design tasks might still bring forth new requirements not considered at the time the schema was conceived. Based on feedback received in the survey, it is suggested to extend the schema in the following directions:

1) support a wider range of methods to execute disciplinary tools (e.g. command line execution on different operating systems)
2) improve the implementation of surrogate models (e.g. their relation with the original tools used to built them, more advanced definitions to store different surrogate model types)
3) include support for a wider variety of workflows: now only MDO architecture-based workflow types coming from KADMOS are supported
4) add measures to reduce the file size
5) extend the documentation now available in the schema repository

Similar to the case of CPACS, a key requirement to facilitate the adoption of a data standard is the availability of a set of supporting applications in its ecosystem. The CMDOWS ecosystem now includes KADMOS and VISTOMS to facilitate users inspecting, reading, visualizing, and editing (in a way that is valid based on the schema definition) CMDOWS files. This means that, as CMDOWS evolves, the whole ecosystem should be extended and kept on a par with the future versions of the schema.

D. KADMOS as platform to formulate MDO systems from tool repository to solution strategy (C4)

As depicted in Figures 4 and 5, KADMOS has successfully played its role as formulation platform for MDO systems, from tool repository to MDO solution strategy (see Figure 2), for a variety of aircraft configurations. The first step where KADMOS comes into play (Step II) is also one of the most appreciated capabilities of the platform by AGILE users: to import the I/Os of a set of CPACS-compatible tools and inspect the couplings between them. In combination with the visualization by VISTOMS, this enables the design team to correctly connect their disciplinary analyses through CPACS, especially when many elements (>4000) from CPACS are involved. Also in the subsequent steps of the MDO problem and solution strategy formulation, KADMOS was found to be a valuable component to the AGILE framework, acting as the "engine" behind the cockpit that what provided by KE-chain in the formulation phase of the five-step approach.

The strong link between KADMOS and CMDOWS has also given KADMOS the secondary role as a library of functions to handle CMDOWS files. This role of KADMOS is extensively used within KE-chain and VISTOMS to read and edit CMDOWS files. Other applications that read CMDOWS files are the PIDO platforms that parse the CMDOWS files as executable workflows. This aspect is more elaborately discussed in Section IV.F. Although the PIDO platforms

only handle a CMDOWS file, the intelligence to formulate a valid MDO solution strategy comes from KADMOS and the KADMOS/CMDOWS combination can therefore be seen as the first half of the bridge between the formulation of MDO systems and their execution (where the parsing by the PIDO platform forms the second half of the bridge). In conclusion, the integrators of the design tasks experienced that KADMOS makes the task of formulating heterogeneous, distributed MDO systems in a collaborative environment much less cumbersome, with an estimated time reduction of 50%. Three main limitations were identified for having KADMOS as the MDO system formulator in the framework.

1. No user friendly interface / interaction

KADMOS is a Python package and does not have its own GUI. In the AGILE MDO framework a user interface was built in KE-chain, where four different forms are used to progress the MDO system definition from tool repository to MDO solution strategy in Step III, see Figure 2. Based on the form input provided by the user, different KADMOS system manipulation methods were executed. However, this way of implementing a KADMOS user interface was found to be too inflexible for some of the peculiarities encountered in the design tasks. This was solved temporarily by enabling the inclusion of custom KADMOS scripts on the platform, although the presence of such scripts dramatically lowers the user friendliness of the interface. The envisioned way forward is to use VISTOMS as KADMOS user interface, as discussed in the next section.

2. Inflexibility of imposing MDO architectures performed by KADMOS

To enable the high level of automation in the formulation phase, a certain inflexibility was implemented in the way KADMOS imposes MDO architectures on MDO problems. The current approach permits a strict boundary between the MDO problem and the solution strategy, where different architectures (e.g. MDF, IDF) can be used to get different strategies for solving the same problem. This approach results in a low amount of settings to be provided by the user, but at the same time it is difficult to set up a more custom-built approach. For example, the design team might want to vary some design variables using a DOE and then optimize the design based on another set of variables. Such a hybrid DOE-MDO architecture is currently not provided in KADMOS and though this specific option could be included with little effort, the addition of all these "mix-and-match architectures" would make the list of architecture options grow tremendously.

Instead of adding (and supporting) all these options, the way forward is to have an advanced mode in KADMOS for users to specify custom-made architectures. Related to such more intelligent architectures, another strong wish in the survey was to include more intelligent analysis of the tool repository. For example, KADMOS could suggest the selection of tools balancing fidelity level and execution time, or the best way to cluster a set of tools to build a surrogate model first, and then executing an optimization using that model.

3. Expected rigidness of tool I/Os (on project and configuration level)

A final, rather detailed, limitation touches upon the core of the AGILE paradigm: the use of CPACS for tool integration in combination with the tool I/O definition required by KADMOS. This limitation is visually explained in Figure 11 and has to do with the fact that, to be able to support formulation, KADMOS has to be very strict in the description of a tool's I/Os, to make sure they are mapped correctly. Hence, KADMOS requires the I/O definition to include the precise definition and amount of certain CPACS elements, while CPACS-compatible tools can usually handle a larger variety in the I/O files and can still be executed. For example, consider that all spars of the main wing are input of a tool. In case two spars are defined in the CPACS file, KADMOS needs to get the following input element definitions, see also the first row of Figure 11:

- /cpacs/vehicles/model/aircraft[@uID="Boeing-747"]/wings/wing[@uID="main_wing"]/spars/spar[1]
- /cpacs/vehicles/model/aircraft[@uID="Boeing-747"]/wings/wing[@uID="main_wing"]/spars/spar[2]

Hence, the input definition specifically states that the aircraft is the Boeing-747, the wing has to be the main wing and then the first and second spar elements are used. The input definition has to be very specific, since the CPACS file might contain other aircraft and/or other wings (i.e. horizontal and vertical tails) for the Boeing-747, and their spars might then be coupled illegally to other tools.

Unfortunately, because of this required strictness, I/O issues arise at two levels: geometrical reconfiguration and project-to-project sharing. With the strict I/O definition, it is not possible to change the amount of spars inside KADMOS without explicitly changing the input file. Hence, if the geometrical configuration (amount of spars) changes in the workflow based on some variable, then this might lead to inconsistencies and errors, see second row in Figure 11. In

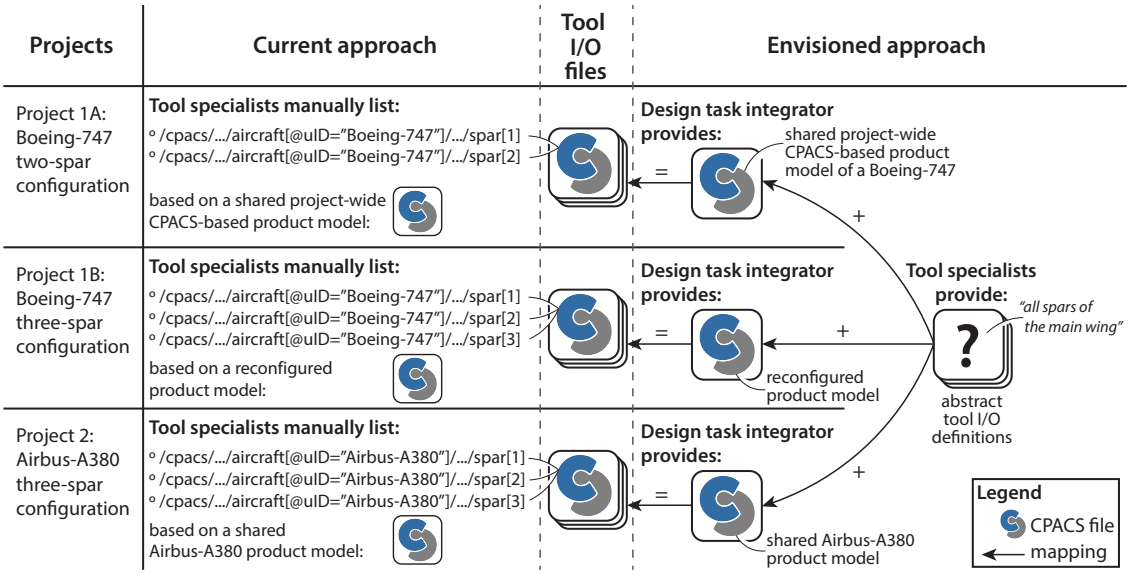

Fig. 11 Visual explanation of the current approach for defining tool I/O files and the envisioned approach for a more flexible definition using a baseline file mapping

addition, also between projects tool I/Os can only be shared if the exact same aircraft (i.e. Boeing-747) is used in the other project as illustrated in the last row of Figure 11. Of course, the uIDs could be replaced, but the essential point here is that a more abstract I/O definition would be required for a more flexible formulation process, such that tool I/Os are defined strictly enough for KADMOS to establish valid couplings, while at the same time geometrical reconfigurations or project-to-project sharing are handled automatically. Practically, this would mean that the earlier spar input definition would not be explicit CPACS elements, but something along the line of: *all spars of the main wing of the aircraft*. Thus there needs to be a mapping of abstract I/O definitions to specific elements of the CPACS-based product model used in a project. The product model is then a single file that has to be adjusted, instead of having to adjust the full collection of I/O files.

Hence, to improve the use of tool definitions between projects and with geometrical reconfigurations, an enrichment layer is required on top of the common language that is already provided by CPACS. This layer would allow a tool I/O definition that is independent of the design task and enable the automatic population of I/O files for a specific task using a mapping approach with the product model. This envisioned approach is visualized in the last column of Figure 11 and could also be used to create a way of project-to-project tool information sharing in KE-chain.

E. VISTOMS as visualization tool for the design team (C5)

Within the scope of the four design cases presented in this paper, VISTOMS has proven to be a valuable visualization tool. The survey indicated that its main strength is the ability to inspect and debug MDO systems interactively in a user-friendly, web-based environment, thereby unravelling the complexity of those systems and keeping the design team well in control of all the system intricacies. This holds especially for large and heterogeneous systems, where the system integrator is usually not the same person as the experts of the individual design and analysis tools. From the survey, as well as from the continuous feedback received during the project, these are the main points for further development, where some concern the current visualization capabilities, some the addition of new capabilities:

1) For systems with very large and deep data schema the tree viewer that is used to navigate (see Figures 6 and 8) to specific CPACS elements becomes somewhat cumbersome. A suggested solution is to alternatively show the elements in a searchable list and thereby provide access to elements based on search strings.
2) Improve the package's handling of multilevel formulations by allowing grouping and compressing of nested subworkflows. Currently, only a single workflow level is possible and this reduces the system overview that VISTOMS is supposed to provide.
3) Although VISTOMS was initially developed purely as a visualization tool, it is suggested to further develop VISTOMS as a full-fledged GUI for KADMOS, thereby enabling also the interactive manipulation of MDO

systems. This development has already started and new functionality will be released in the near future.

4) Include a debugging mode in which the results of the graph analyses performed by KADMOS can be directly rendered in the VISTOMS visualization. These analyses include typical problems for MDO systems, such as multiple tools writing the same CPACS element (collision), or tools without any I/O connections.
5) Once the connection between VISTOMS and KADMOS is made, the resulting package could be easily linked to the open-source framework OpenMDAO[20] via the OpenLEGO package, to act as a GUI for building collaborative MDO systems in a fully Python-based environment. In this way KADMOS, VISTOMS and OpenLEGO could become all part of the already popular OpenMDAO ecosystem.
6) Set up an open-access website where people can use VISTOMS. The visualization tool is already open-source and its code is part of the KADMOS repository. However, at the moment, the tool needs to be deployed manually on a local system. A website, where anyone can start an interactive VISTOMS session which would also contain the KADMOS and CMDOWS components, would lower the threshold for interested users to quickly explore key components of the AGILE paradigm.

The survey indicated that the use of VISTOMS is estimated to provide time savings of almost 40% for the setup of collaborative MDO systems. The four design tasks provided valuable feedback to test and debug VISTOMS. In combination with the feedback collected in the survey, a clear direction for the future development of the package has been established.

F. RCE as CMDOWS parser to create executable workflows (C6)

All the CMDOWS files of the formulations shown in Figures 4 and 5 have been successfully parsed as executable workflows in RCE. One example from the box-wing aircraft design task is shown in Figure 10. Being one of the latest AGILE outcomes, the RCE CMDOWS parser is the least mature component of the automation approach. The three most critical issues, as emerged from the survey, are discussed below.

1. Large overhead due to data handling

The CPACS-based tool integration in the AGILE paradigm has major implications on the data handling in the workflows. Not all elements in RCE (but also in other PIDO platforms) are built to have XML files as I/Os. Therefore, XML readers and writers are required around these components to integrate them in the workflow, see for example the optimizer and converger elements in Figure 10. Similar data handling has to be done when two tools are executed in parallel and their CPACS output files need to be merged (also shown in Figure 10). This continuous data translation adds a large overhead to the execution time of the workflow.

2. Lazy data handling approach leads to inconsistent CPACS files

In order to reduce the overhead discussed above, a "lazy data handling" approach is implemented in RCE, which means that CPACS-compatible tools are assumed to read a CPACS file and then append their results to the same file. This way, in a sequential execution of CPACS-compatible tools, the files can be passed directly without data handling overhead, see the tools in the converger loop in Figure 10. However, when tools are executed in parallel then their results need to be merged as one CPACS file if the next tool requires results from both tools, as is the case for the two tools in the lower right corner of the RCE workflow depicted in Figure 10. Since data of more upstream tools have been added to the CPACS files of the parallel tools and were forwarded indirectly based on the lazy data handling, the correct merging of the files is not always evident. There is the risk that a value was updated by a tool further upstream of one parallel tool, that might be ignored if the merging of the files takes the old value from the other parallel tool.

These data handling issues could be resolved by implementing a stricter data handling for parsed workflows, though such data handling will also have a negative impact on the already large overhead involved. As was discussed in the previous section, KADMOS is aware of the exact I/Os of a tool and this information is also stored in the CMDOWS file used for parsing. Therefore, the information could be used in the RCE workflows for data handling, among others to assure the consistency of the CPACS files being generated and merged. At the same time, the data handling should be less strict when possible to reduce overhead. Thus, the parsing method should be a trade-off between strict data handling for CPACS consistency and overhead reduction.

3. Hidden assumptions in the parsing approach

The last main limitation brought forward in the survey is related to the many hidden assumptions that are part of the parsing process. When workflows are built manually, the integrator usually adapts the workflow to specific needs of different tools and no specific requirements for tool integration exist. For example, some tools might simply provide the outputs that they have calculated and then the integrator would manually set up a merge of those outputs with the full CPACS file to append the information. But in the AGILE paradigm tools are supposed to be integrated by appending information to the CPACS file directly and if that has not been done, then the workflow cannot execute correctly. Hence, in order to automate workflow creation, the tool integration has to meet specific requirements for the automation to work correctly. These kind of requirements are sometimes difficult to find, especially if the user is not aware of key assumptions that are part of the parsing process. In this sense the automation approach in Figure 2 is clearly part of an distinct paradigm, meaning that it requires users to be aware and adapt to core assumptions of its components to perform their automation task correctly. This threshold for adopting the paradigm could be lowered by clearly communicating its key assumptions and checking user's input more elaborately in that respect.

An average time reduction of 33% was estimated by the expert users of RCE in the survey. At the moment CMDOWS parsers have been created for three PIDO platforms: Optimus, RCE and OpenMDAO (via OpenLEGO), of which the latter two are open-source. Although the CMDOWS standard has already been adopted by a wider variety of MDO support applications, its true initial development goal was to enable the creation of executable workflows from a neutral workflow definition. This position of CMDOWS could be further strengthened if more commercial PIDO platforms would adopt this, thus creating a virtuous snowball effect leading to a wider adoption of the schema, and possibly to the upgrade of CMDOWS as official standard. The initial feedback from the commercial PIDO tool provider in AGILE is very positive and they consider this approach the way forward to improve their capability to support MDO expert customers with customized deployments of their workflow management system.

G. Complete AGILE MDO framework

The complete AGILE MDO framework was positively reviewed in the survey and AGILE partners estimated that a 40% time reduction for setting up collaborative MDO systems could be achieved. A trend found in the survey feedback was that technically the framework is performing well, but to grasp, use and appreciate it still requires effort. Indeed, the AGILE MDO framework is the technical implementation of a truly novel paradigm to perform MDO, as such it takes a mind shift and some time to get acquainted. As the current version of the framework is technically sound (with only minor bugs and issues to be solved for the different components), it would make sense to reduce the effort for embracing the AGILE paradigm by consolidating the different components and making the GUIs more intuitive, informative and structured. The possibility to provide different look and feel to the different users, according to their specific role in the MDO process, has already been taken in consideration by the KE-chain developers.

Concerning the proposed standard data schemas CMDOWS and CPACS, it was apparent in AGILE how significant their impact is in supporting the integration of a heterogeneous set of disciplinary tools and MDO support applications. However, two main risks are associated with their use:

- The schemas need to keep up with an ever-changing environment, while they tend to loose flexibility when maturing.
- A wider adoption of a schema requires the involvement of more people in its development, therefore a more professionally set up development process is necessary.

As already mentioned, the creation of an international review committee is recommended for the already mature and widely adopted CPACS. In the case of CMDOWS, a broader user base is necessary to justify the setup of a dedicated committee.

Conceptually, the AGILE MDO framework connects two very distinct phases of developing a collaborative MDO system: formulation and execution. A crucial limitation found in AGILE, caused by the use of a central data schema (i.e. CPACS), is that a very strict definition of tool I/Os is necessary for a specific design task in the formulation phase, while tool execution should preferably be based on less strict I/Os so that multiple tools can handle a variety of CPACS files and thereby be used simultaneously in different design tasks. This formulation-execution discrepancy impacts many components, as discussed in their respective sections. As it relates back to the root of the AGILE paradigm, future developments should focus on how to tackle this discrepancy starting at the central data schema that is adopted.

Another topic, that has not been addressed in this paper nor in the AGILE MDO framework as a whole, concerns the use of derivatives to speed up gradient-based optimization strategies. Most of the tools available within the AGILE

consortium were not able to provide derivatives and given the fact tool development was not a goal of the project, it was decided not to include the use of derivatives in the framework. Naturally, including this concept in the framework would impact all six components; derivative values needs to be stored by the tools in CPACS files, their role in the MDO system needs to be stored in CMDOWS files, they need to be visualized in VISTOMS, handled properly by KADMOS to formulate MDO solution strategies and, finally, executable workflows need to be parsed that include optimizers using the derivative information. As derivatives offer a huge potential for enabling large-scale MDO, it is considered to be another crucial topic for future development.

Lastly, survey respondents considered the automation approach supported by the six components to be a valuable addition to the current MDO ecosystem. Future use of the framework was suggested in other collaborative R&D projects (e.g. PARSIFAL) as well as in high-level education (e.g. capstone design projects such as the Design Synthesis Exercise at DUT). Also it was suggested to assess the framework's capabilities in other engineering domains, such as automotive, wind energy and infrastructure.

V. Conclusions

With the AGILE project nearing its completion, a critical look on the automation approach for the formulation and execution of collaborative MDO systems developed over the past two years has been presented. Six key components of the AGILE framework were put to the test in four design tasks concerning the MDO of novel aircraft configurations. A preliminary feedback from the AGILE partners on the developed MDO framework was collected through an extensive survey.

Overall outcome of the assessment is that all six components were considered valuable additions to the MDO ecosystem to effectively operate in a collaborative environment with large and heterogeneous teams of experts. The framework was estimated to provide a time gain of 40% for the setup of MDO systems. The framework, however, was found valuable also for a number of other critical aspects: the lower accessibility provided to a complex discipline such as MDO; the ability to keep control and complete oversight of complex computational systems spanning outside the usual desktop boundaries; the capability to enhance collaboration between specialists normally operating at different levels and on different aspects of the design process. At the same time valuable indications for future developments have been collected. While many were of specific technical nature, one important recommendation is to further seek the balance between providing full control of all the aspects of the MDO system setup and simple guidance to allow the various specialists providing just their own contribution. The AGILE MDO framework is considered a first step in the right direction for the establishment of a 3rd generation framework, as well as an initial definition of the paradigmatic shift required to perform collaborative, distributed MDO accounting for the technical and non-technical hurdles that it may present.

Acknowledgments

The research presented in this paper has been performed in the framework of the AGILE project (Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts) and has received funding from the European Union Horizon 2020 Programme (H2020-MG-2014-2015) under grant agreement n° 636202. The authors are grateful to all partners of the AGILE consortium for their contribution to this paper. In particular, to the DLR software developers Sascha Zur and Tobias Brieden for their work on the CMDOWS parser in RCE and to the design task integrators Luca Stingo (UNINA), Pierluigi Della Vecchia (UNINA), Thierry Lefebvre (ONERA), Dominique Charbonnier (CFSE) and Francesco Torrigiani (DLR) for their continuous and valuable feedback while using the AGILE MDO framework during its development process.

Open-source references

CMDOWS repository	`http://cmdows-repo.agile-project.eu`
CMDOWS interface	`http://cmdows.agile-project.eu`
CPACS repository	`https://github.com/DLR-LY/CPACS`
KADMOS repository	`https://bitbucket.org/imcovangent/kadmos`
OpenLEGO repository	`https://github.com/daniel-de-vries/OpenLEGO`
OpenMDAO repository	`https://github.com/OpenMDAO/OpenMDAO`
Pykechain	`http://pykechain.readthedocs.io/en/latest`
RCE	`http://rcenvironment.de`
TiGL repository	`https://github.com/DLR-SC/tigl`
TIXI repository	`https://github.com/DLR-SC/tixi`
VISTOMS repository	`https://bitbucket.org/imcovangent/kadmos/src/master/kadmos/vistoms`

References

[1] Agte, J., De Weck, O., Sobieszczanski-Sobieski, J., Arendsen, P., Morris, A., and Spieck, M., "MDO: assessment and direction for advancement - an opinion of one international group," *Structural and Multidisciplinary Optimization*, Vol. 40, No. 1-6, 2010, pp. 17–33.

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

[3] Shahpar, S., "Challenges to overcome for routine usage of automatic optimisation in the propulsion industry," *Aeronautical Journal*, Vol. 115, No. 1172, 2011, p. 615.

[4] Ciampa, P. D., and Nagel, B., "The AGILE Paradigm: the next generation of collaborative MDO," *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2017.

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

[6] Ciampa, P. D., Baalbergen, E. H., and Lombardi, R., "A Collaborative Architecture supporting AGILE Design of Complex Aeronautics Products," *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2017.

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

[8] Nagel, B., Böhnke, D., Gollnick, V., Schmollgruber, P., Rizzi, A., La Rocca, G., and Alonso, J. J., "Communication in aircraft design: Can we establish a common language?" *28th International Congress Of The Aeronautical Sciences, Brisbane*, 2012.

[9] van Gent, I., La Rocca, G., and Hoogreef, M. F. M., "CMDOWS: A Proposed New Standard to Store and Exchange MDO Systems," *CEAS Aeronautical Journal*, 2018.

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

[11] Martins, J. R. R. A., and Lambe, A. B., "Multidisciplinary Design Optimization: A Survey of Architectures," *AIAA Journal*, Vol. 51, No. 9, 2013, pp. 2049–2075.

[12] Aigner, B., van Gent, I., La Rocca, G., Stumpf, E., and Veldhuis, L. L. M., "Graph-based algorithms and data-driven documents for formulation and visualization of large MDO systems," *CEAS Aeronautical Journal*, 2018.

[13] Lambe, A. B., and Martins, J. R. R. A., "Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes," *Structural and Multidisciplinary Optimization*, Vol. 46, No. 2, 2012, pp. 273–284.

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

[15] van Gent, I., Lombardi, R., La Rocca, G., and d'Ippolito, R., "A Fully Automated Chain from MDAO Problem Formulation to Workflow Execution," *EUROGEN 2017*, 2017.

[16] Baalbergen, E., Kos, J., Louriou, C., Campguilhem, C., and Barron, J., "Streamlining cross-organisation product design in aeronautics," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 231, No. 12, 2017, pp. 2192–2202.

[17] Jepsen, J., Ciampa, P. D., and Nagel, B., "Avoiding inconsistencies between data models in collaborative aircraft design processes," *65. Deutscher Luft- und Raumfahrtkongress*, 2016.

[18] La Rocca, G., "Knowledge based engineering techniques to support aircraft design and optimization," Ph.D. thesis, Delft University of Technology, 2011.

[19] de Vries, D., "Towards the Industrialization of MDAO," Master's thesis, Delft University of Technology, 2017.

[20] Gray, J., Moore, K. T., and Naylor, B. A., "OpenMDAO: An open source framework for multidisciplinary analysis and optimization," *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, Vol. 5, 2010.