Distributed Multidisciplinary Optimization and Collaborative Process Development using RCE

Brigitte Boden[1], Jan Flink[2], Robert Mischke[3], Kathrin Schaffert[4], Alexander Weinert[5], Annika Wohlan[6]
German Aerospace Center (DLR), Simulation and Software Technology, 51147 Köln, Germany

Časlav Ilić[7], Tobias Wunderlich[8], Carsten Liersch[9], Stefan Görtz[10]
German Aerospace Center (DLR), Institute of Aerodynamics and Flow Technology, 38114 Braunschweig, Germany

and
Erwin Moerland[11], Pier Davide Ciampa[12]
German Aerospace Center (DLR), Institute of System Architectures in Aeronautics, 21129 Hamburg, Germany

DLR's Remote Component Environment (RCE) is an open-source software environment for defining and executing workflows containing distributed simulation tools by integrating them into a peer-to-peer network. It is being developed primarily by DLR and has been used in various engineering projects, including several aerospace projects dealing with multidisciplinary optimization (MDO) and multidisciplinary analysis (MDA). RCE has several advantages that can help to achieve more reusable multidisciplinary processes. The workflow is composed of built-in and user-defined components. Disciplinary tools are integrated as standalone components, with defined inputs and outputs, and then distributed over the network. While executing the workflow, data dependencies between the components are automatically detected, and a component is executed as soon as all its input data is available. Thus, multiple components can run at the same time. The components of a multidisciplinary process can also be executed in a distributed manner, where the tools are located on different machines with possibly different operating systems. Once configured, the peer-to-peer network is automatically established between the RCE instances running on different machines, making components visible and executable even between instances that are only connected indirectly. The distributed execution capability alleviates tool deployment issues, including those related to the protection of intellectual property. RCE supplies a graphical editor for creation of workflows, using the built-in components to control the data flow. Some built-in components can be used to perform optimization tasks within the workflow, including nested loops, using built-in or user integrated optimization algorithms. We demonstrate the use of some of the key features of RCE for MDA and MDO purposes based on several collaborative DLR projects where distributed simulation tools are being used.

[1] Research Engineer, Distributed Software Systems Group, Intelligent and Distributed Systems Department.
[2] Team Leader, Distributed Software Systems Group, Intelligent and Distributed Systems Department.
[3] Team Leader, Distributed Software Systems Group, Intelligent and Distributed Systems Department.
[4] Research Engineer, Distributed Software Systems Group, Intelligent and Distributed Systems Department.
[5] Research Engineer, Distributed Software Systems Group, Intelligent and Distributed Systems Department.
[6] Research Engineer, Distributed Software Systems Group, Intelligent and Distributed Systems Department.
[7] Research Engineer, MDO Group, Transport Aircraft Department.
[8] Research Engineer, MDO Group, Transport Aircraft Department.
[9] Research Engineer, Conceptual Aircraft Design Group, Transport Aircraft Department.
[10] Team Leader, Aerodynamic Surrogates and Optimization Group, C²A²S²E Department, AIAA Member.
[11] Team Leader, Collaborative Engineering Group, Aircraft Design & System Integration Department.
[12] Team Leader, MDO Group, Aircraft Design & System Integration Department.

I. Introduction

The analysis, design, and simulation of complex systems such as aircraft require the use of a multitude of software tools from a wide range of disciplines. While each disciplinary tool focuses on solving a single sub-problem, solving the overall aircraft design problem using optimization algorithms significantly increases the number of tools involved and the overall complexity and usually requires leveraging distributed computational resources. RCE (Remote Component Environment) is an open-source distributed integration framework primarily being developed at DLR [1,2,3].Both the source code of RCE as well as executable versions for Windows and Linux are available free of charge under a permissive license [4]. RCE helps engineers and scientists to analyze, optimize, and design complex systems like aircraft, ships, or satellites. Handling complex systems requires many experts and several tools. RCE is especially suited for multidisciplinary collaboration and concurrent engineering. Its target audience comprises engineers and scientists who are involved in multidisciplinary design and optimization processes comprised of numerous disciplinary software tools. As such, RCE is a software framework that facilitates setting up and solving distributed MDA and MDO problems. RCE enables these users to construct executable descriptions of such processes, to execute them in a distributed environment, and to monitor and analyze all data accrued during execution of the process.

Previously, RCE was successfully deployed in various aerospace projects [5,6] and for engineering problems in the shipbuilding, energy and transport sectors. The purpose of this paper is to showcase RCE's capabilities, recent improvements, and some new demonstrative applications in aeronautics.

The paper is organized as follows: RCE with its unique features for distributed execution of tools and collaborative process development are described in detail in the next section. Next, we briefly introduce the Common Parametric Aircraft Configuration Schema (CPACS), a hierarchical XML file which is used for the formalized exchange of product data in aircraft design. Then, we present the use of RCE for MDA in two collaborative aircraft design projects. Finally, we showcase how RCE is being used in conjunction with CPACS for distributed MDO in three different collaborative projects.

II. Remote Component Environment (RCE)

In order to describe a process chain in a way that can be executed automatically, RCE asks the user to first define the interfaces of the software tools involved in the process. The user can subsequently compose these tools into the complete process, the so-called workflow. In this workflow, the tools are merely viewed as black-boxes comprised of inputs and outputs, but no description of their internal behavior. Afterwards, the user can execute the resulting workflow via RCE. For each tool involved in the process they may choose a machine on their network on which to execute the tool. RCE transparently handles the transport of data between the machines involved in the execution of the workflow and retains the accrued data for later analysis.

In this section, we first describe the general principles behind the definition of such black-boxes, their composition into workflows, and the execution of such workflows on a single machine. Afterwards, we continue by highlighting some main features of the graphical user interface for managing and administrating RCE. We then detail how multiple instances of RCE can be composed into a network for distributed execution of workflows as well as the additional challenges encountered in such a setting. Finally, we close with a brief outlook on the further development of RCE.

A. Tool Integration, Workflow Construction and Execution

The primary use cases of RCE consist of the integration of external software tools into the integration environment, the composition of such tools into workflows, the execution of these workflows, and the retrieval of data accrued during this execution. In order for tools to be integrated into RCE they only have to meet two requirements: They must (a) be executable from the command line and (b) must not require any user interaction during execution.

Integrating an external tool into RCE amounts to defining an interface between RCE and the tool, i.e. defining its inputs and outputs to make them accessible by RCE, adding pre- and post-processing steps for the input and output data, and defining the commands for the invocation of the tool. In order to aid the user in defining this integration, RCE features a graphical wizard that guides the user through this process. Once created, the integration is defined by a plain text file which can also be edited using standard text editors. An integrated tool is available locally as a component to be used in workflows and fits seamlessly into the user interface.

In addition to user-integrated tools, RCE provides a number of predefined tools which can be used in conjunction with integrated tools to construct complex workflows. These predefined tools supply a multitude of basic functionalities used in numerous workflows such as handling the flow of data through the workflow, reading and extracting data from files, manipulating XML, executing user-defined Python scripts, and evaluating incoming data. Furthermore, there are predefined components that simplify the construction of workflows for multidisciplinary design optimization, such as basic mathematical and statistical methods. There is also a component that determines absolute or relative convergence of its input values, and a component that provides access to the optimization algorithms implemented by the Dakota software library. Moreover, RCE features a component that allows for the exploration of a parametric design space. This component provides several algorithms for this exploration, among them a design based on Latin Hypercube sampling or on a Monte Carlo approach. The user, however, also has the flexibility to specify a custom design.

After integrating the tools required for the execution of the workflow, the user may compose them into a workflow. To this end, RCE offers a graphical editor allowing the user to construct a workflow by first dragging and dropping the required components into the editor and subsequently connecting their respective inputs and outputs.

After constructing such a workflow, the user can execute it. RCE starts by executing those tools that have no inputs, e.g., tools that merely read input files from some preconfigured location. Once a tool has terminated, e.g., once it has read a file into memory or finished its simulation, RCE collects the outputs of the tool as defined by its integration and computed by its post-script and passes them to the tools whose inputs are connected to the outputs of the terminated tool. After forwarding all inputs required by a given tool, RCE executes the respective tool and again collects its outputs in order to pass them to subsequent tools. The execution of the workflow continues in this manner, with several tools potentially running concurrently, if their input and output data do not depend on one another. During execution of the workflow, RCE moreover provides the user with monitors for the progress of the workflow as a whole as well as for the progress of individual tools. In the following section, we provide further detail on the graphical interface of RCE that aids the user in integrating tools as well as in constructing and executing workflows.

B. User Interface

Since it is the goal of RCE to support users in their daily tasks, we aim to keep the barrier to using RCE as low as possible. Furthermore, since RCE is not developed as a single-purpose software, but rather as a flexible distributed integration framework, we aim to provide easy access to all of the features provided by RCE. To accomplish both goals simultaneously, RCE offers not only a customizable graphical user interface, but also a command line interface and an embedded SSH server that allows for remote control and monitoring. In this section, we focus on a description of the graphical user interface before briefly touching upon the text-based interfaces in Section “Tool Publication and Networking”.

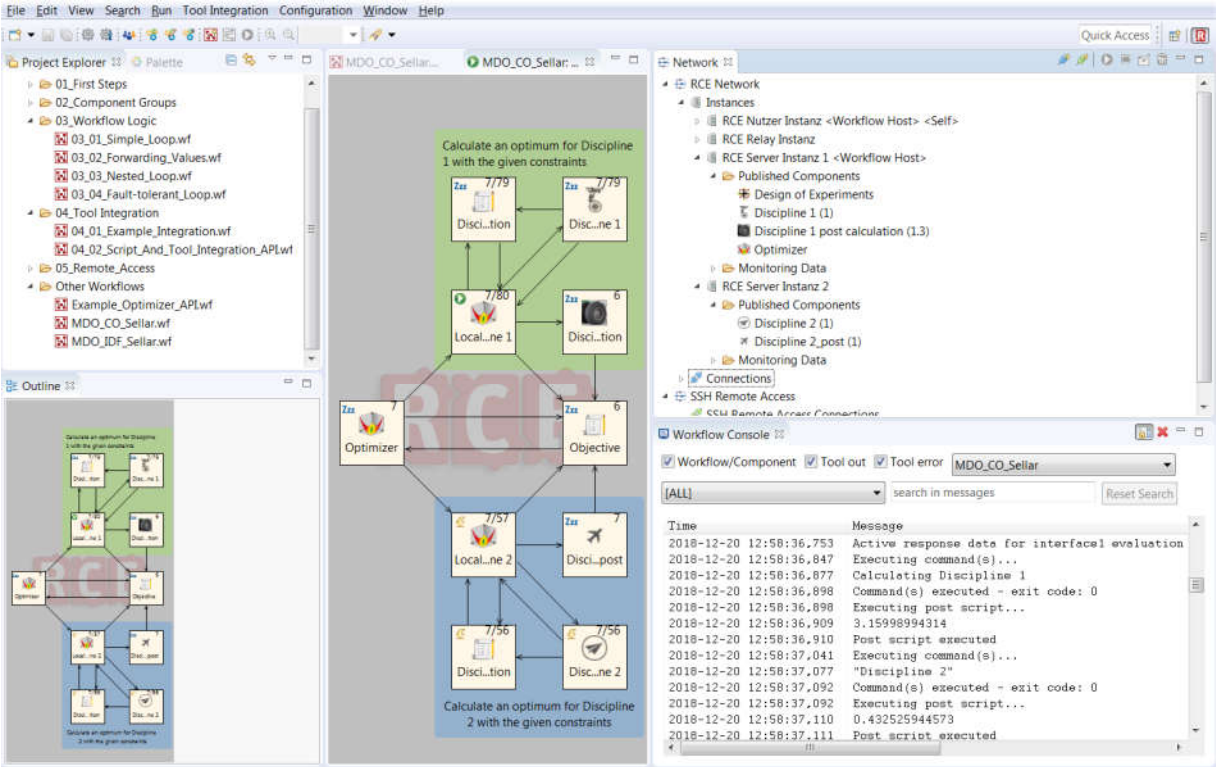
For many users, the first step when using RCE is the integration of a custom tool. As explained above, this step involves the definition of an interface between the tool and RCE, as well as the implementation of pre- and post-processing steps and the definition of commands for the invocation of the tool. This process is, in general, quite involved, as it requires precise and complete knowledge about the input and output formats of the tool as well as the implementation of scripts for parsing the tool's output. In order to simplify this process as much as possible, RCE provides a graphical wizard that guides the user through the integration of a tool. This wizard presents the process of integrating a tool as a number of discrete and independent sub-steps, each of which is independent of each other. Hence, integrating a tool using this wizard is easier by far than manually writing a text file defining all parts of the integration at once.

One of RCE's main features consists of giving users the ability to compose tools into workflows. In order to assist users with this, a major part of RCE's main user interface is comprised of a graphical editor for the visual creation of workflows. Using drag and drop, users can intuitively add the desired workflow components and arrange them in a logical sequence. After thus arranging the tools, the user can easily connect them via a connection manager which displays the flow of data between two selected components. Again leveraging drag and drop, the user can connect the inputs and outputs of the components. To further improve usability, the connection manager automatically connects the inputs and outputs with the same name and the same data type. We show such a completed workflow as displayed in RCE in Fig. 1.

Furthermore, we have observed in practice that users rarely work on a single workflow, but are rather involved in the construction, execution, or analysis of multiple workflows, sometimes even among multiple projects. In order to account for such scenarios, RCE features a graphical project manager which gives an overview over several workflows that the user may work on concurrently, together with other resources that they may require for the execution of such workflows, such as input data or template files. This project manager can be seen in the top left corner of Figure Fig. 1.

Finally, we have observed that the adoption of RCE used to be greatly impeded by the disconnect between RCE and its documentation. New users regularly had to reference an external guide during their initial hours with RCE, which used to be provided only as a separate PDF file. In order to alleviate this, RCE also features a context-sensitive online help, which is easily accessible by the user and displays documentation that is relevant to the current situation and environment of the user.

To conclude, RCE does not only feature a graphical user interface for those tasks that necessitate such an interface, such as the construction of workflows, but for a wide range of supplementary tasks, such as tool integration or project management. While a graphical interface is not necessarily required in order to accomplish these tasks, they have shown to greatly simplify the everyday use of RCE. We now proceed to discuss the use of RCE in a distributed setting, where several tools that are combined in the same workflow are executed on multiple physical machines in the same network.



```
Time                     Message
2018-12-20 12:58:36,753  Active response data for interface1 evaluation
2018-12-20 12:58:36,847  Executing command(s)...
2018-12-20 12:58:36,877  Calculating Discipline 1
2018-12-20 12:58:36,898  Command(s) executed - exit code: 0
2018-12-20 12:58:36,898  Executing post script...
2018-12-20 12:58:36,909  3.15998994314
2018-12-20 12:58:36,910  Post script executed
2018-12-20 12:58:37,041  Executing command(s)...
2018-12-20 12:58:37,077  "Discipline 2"
2018-12-20 12:58:37,092  Command(s) executed - exit code: 0
2018-12-20 12:58:37,092  Executing post script...
2018-12-20 12:58:37,110  0.432525944573
2018-12-20 12:58:37,111  Post script executed
```

Fig. 1. The graphical user interface of RCE showing, among others, the workflow editor. The yellow boxes denote disciplinary tools, while the arrows between them denote the exchanged data

C. Tool Publication and Networking

Many projects in the context of research and engineering require the connection of tools between different organisations. Thus, RCE does not restrict the user to those tools integrated on their local machine when constructing a workflow. Instead, they can connect their local instance of RCE with instances running on remote machines reachable via a local network or the internet. RCE's workflow editor transparently includes the tools integrated on those remote machines, allowing the user to include them in the workflow as if they were integrated locally. The setup of connections to remote machines is again aided by a graphical user interface that displays the existing connections as well as all instances currently visible on the network.

By default, connections are unencrypted and as such only meant to be used only in a trusted (internal) network. In order to allow communication across the boundaries of organizations, RCE also provides a different type of connections, which utilize the SSH protocol to provide encryption and login authentication. These connections are meant to securely connect RCE instances over the internet.

In this section, we first explain how users can leverage the network of RCE instances to execute workflows in a distributed fashion. Subsequently, we demonstrate RCE's versatility for use in such a distributed environment in which the graphical user interface explained above may not be available. Finally, we detail challenges that arose from the ever-increasing size of RCE networks in practice and recent improvements in the architecture of the handling of network connections we made in order to overcome these challenges.

1. Distributed Execution and Authorization

In very simple scenarios, it is possible to grant all connected instances universal access, i.e., to allow each instance to access all integrated tools on all other instances. In practice, however, it turns out that this model is often undesirable. On the one hand, this permissive model would lead to cluttering of the user interface, as users are presented with numerous remote tools that are unnecessary for their work. On the other hand, if a user is a member of different RCE networks at the same time, it may be necessary to restrict certain tools to one network, but prevent them from being accessed from the other.

Hence, in order to obtain more fine-grained control over the accessibility of the published components, each component can be assigned one of three publication levels: By default, a component is published *locally* and can only be accessed by the local instance. If a user wants to grant universal access to a tool, they may assign it the publication level *public*, which exposes that tool to all connected RCE instances across the network. As a middle ground, the *custom* publishing level grants access to the component to only to instances that are members of certain authorization groups. These authorization groups are defined by the user publishing the tool. In order to grant access to a specific authorization group, the user defining the group must distribute an automatically generated secret key to the appropriate users, or in case of server machines, to their administrators.

When constructing a workflow, the users can transparently include tool integrations published by remote instances of RCE. Such remotely integrated tools are indistinguishable from locally integrated ones in the graphical workflow editor. This separates the construction of the workflow logic from the decision on which machine to invoke the required tools, thus greatly simplifying the former task.

Software used in multidisciplinary design and optimization processes often requires very specific runtime environments, e.g., due to required hardware or due to licensing requirements. Hence, when executing a resulting workflow comprising both local and remote components, the latter ones are not copied to the machine executing the workflow, but they are instead executed on the machine on which they are integrated: RCE supplies them with the required input data via the network connection, retrieves their output after termination and returns it to the machine executing the workflow. Due to this approach, users are not required to locally replicate the environments required for executing disciplinary software, but can instead leverage existing infrastructure that is tailored to the execution of the individual, disciplinary tools.

Finally, the coordination of the workflow itself may be passed to a remote machine: Upon execution of a workflow, the user may designate some instance on the network to be the so-called workflow controller, which orchestrates the distribution of data among the involved instances of RCE. This simplifies the execution of long-running workflows, as the user may start the workflow via their local RCE instance and designate a server as the workflow controller, allowing them to shut down their local machine after the start of the workflow.

In order to determine the correct behavior of a workflow, users often not only examine the final results, but also require intermediary data that is passed between the disciplinary tools. Hence, all data accrued during a workflow run, i.e., all inputs and outputs to all components are collected by the workflow controller. Users can monitor this data both during execution of the workflow and after its completion by all instances connected to the workflow controller. This provides a huge benefit for the collaborative analysis of the results of the workflow over the existing ad-hoc dissemination of such results, which usually occurred via email or shared network folders. Among other features, RCE offers, e.g., a timeline to the user, which displays the start- and end-time of the execution of each tool and allows the user to investigate the sequence of tool invocations in their workflow. We show an example of such a timeline in Fig. 2.

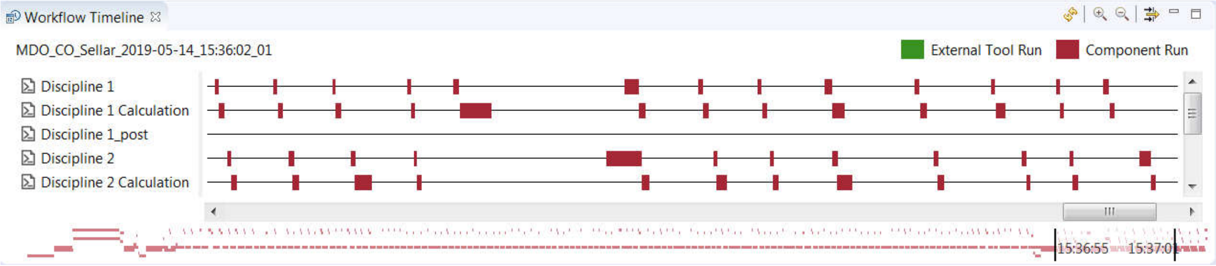


Fig. 2. The timeline of an RCE workflow execution. The red bars denote the execution of a single disciplinary tool.

2. Deploying RCE on a Server

In the previous section, we have discussed the use of RCE in a distributed environment comprised of both user-facing machines as well as machines only configured for remote access via a text-based interface. When used directly by a user, the instance is installed on the personal computer of a user and provides the graphical user interface described above in order to facilitate the integration of tools and the construction of workflows as described above. When deployed on a server, in contrast, RCE usually serves to publish and execute tools installed on that server. In order to support this latter use case, RCE can be started without its graphical user interface and is configurable via a command line interface. While this is less intuitive than the configuration via the graphical user interface, it allows for the use of RCE on remote servers. In order to further ease usability on remote servers, this command line interface is not only available locally, but it is also exposed via SSH. Hence, users can use the SSH client of their choice to directly connect and administer a remote RCE instance instead of first having to connect to the machine that RCE is deployed on. Furthermore, RCE can be installed as a background service/daemon (on Windows/Linux machines, respectively), in order to further facilitate its use in this context.

Finally, we have observed that in practice, the machines involved in the design and execution of a workflow may not necessarily be located within the same logical network. A frequent example is that the users or server machines within a project may be located in different organizational units which are prevented by network firewalls from directly connecting to each other. To still allow direct collaboration, RCE may serve as a relay server, which forwards all received communication to all of its remotely connected machines, acting as an intermediary that simplifies the process of connecting these machines and users. In effect, this allows the creation of overlay networks of RCE instances comprising numerous machines and spanning multiple physical networks.

3. Resilience and Robustness of Workflow Execution

In recent years, the workflows constructed using RCE have grown in size and scope. Early users constructed workflows with at most around a dozen components which were executed on one or two machines within at most an hour. Contemporary workflows, on the other hand, feature hundreds of components that are executed on dozens of machines and may take several weeks to complete. This increase in scope brought with it an increased amount of communication between the involved machines during workflow execution. This, in turn, caused issues with the involved machines or the underlying network to surface more frequently during the execution of a workflow. As such issues often led to the termination of the workflow, we recently focused on making RCE robust against such issues.

Since RCE 9.0, workflow execution is robust against typical causes of workflow disruption. The low-level foundation of this robustness approach is a mechanism that stores and repeats network messages when necessary. For request-response message cycles, which are widely used in workflow execution, such a mechanism must cover three failure cases: Errors during the transmission of the request; errors while the request is being processed by the receiver; and errors during transmission of the response back to the sender. All of these are handled transparently by the current implementation. We illustrate how this mechanism improves workflow-level robustness by considering typical failure modes that caused workflows to abort before these changes were introduced, namely network overload and host overload. One of the most frequent causes of workflow failures used to be temporary losses of network connectivity, or network overload situations. When a workflow is running for weeks, experience shows that these can occur even in apparently stable networks. In case of such a network disruption, the retry mechanism detects the failure to transmit one or multiple network messages, and monitors the network for the target machine to become reachable again. Once this occurs, the mechanism verifies that this machine is still ready to participate in the workflow, and then resends the lost messages. The same principle is applied when a machine that received a network request detects that transmitting its response has failed.

Another frequent cause of workflow failures used to be that the recipient of a network message is unable to respond to it in a timely fashion, typically because of CPU or I/O overload. As each RCE instance runs on the same machine that is used to execute the actual integrated tools, and since many of these tools use all available computing power by default, such a situation can easily occur. In these cases, network messages are typically still transmitted, but processing them takes unusually long. This is detected by the sender as a timeout, which causes them to re-transmit the request. The receiver receives the request again, notices the duplication, and proceeds with the original execution. Once this has finished, a normal response is generated and sent back, which the original sender associates with the repeated request. This causes the calling machine to proceed as if no timeout had occurred in the first place.

This illustrates that starting with version 9.0, RCE is robust against intermittent failures both during the transport of messages between two instances, as well as against high load on the involved machines. Together with the possibility for fine-grained publication of tools to remote instances as well as its versatility for the use in both user-facing and server environments, RCE is well suited for the distributed execution of multidisciplinary workflows comprised of widely disparate tools with heterogeneous runtime requirements.

4. Cross-organizational Collaboration

We are currently working on extending the capabilities of RCE for cross-organization, encrypted connections. While it is already possible to encrypt the communication between two RCE instances using the SSH protocol as stated above, these connections are currently limited to point-to-point connections, and do not support the full set of features provided by standard connections yet. In current work, we are aiming to support a larger feature set, while still constraining these features to ensure that only the intended data and tool access is provided to external users. Monitoring and auditing features may also be created as needed. Another important upcoming feature is support for connections between different organizations which are mediated by a so-called relay server, functionally similar to the relay servers for internal networks mentioned above. In addition to simplifying multi-organization project setups, this feature will also address frequent IT security concerns as it eliminates the need to provide any open network ports, or any externally reachable service, except for a single institution providing the relay server. All data sent across this relay server will be additionally encrypted to ensure that even the relay's administrators cannot access transmitted simulation data without permission.

III. Common Parametric Aircraft Configuration Schema (CPACS)

The use of a common product data model significantly reduces the amount of interfaces between the different tools and components and establishes a common language for communicating product and tool specific knowledge. This is illustrated in Fig. 3. Since 2005, DLR has developed the Common Parametric Aircraft Configuration Schema (CPACS) [7,8,9]. The schema provides a standardized structure for the formalized exchange of product data within aircraft design. The hierarchical structure of the schema definition allows the consistent exchange of product data on multiple fidelity levels. Support in the connection of legacy tools to the central product data model is provided by the XML and geometry interface libraries [10,11], providing the disciplinary engineering services capabilities for querying information from CPACS files.

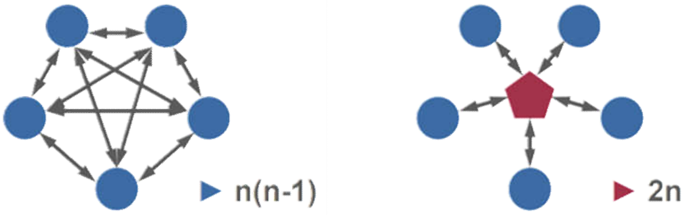


Fig. 3. Significant reduction of the amount of interfaces between engineering services by adopting a common product data model

Nowadays, CPACS is becoming a de-facto standard for the exchange of product data within preliminary aircraft design. Its continued development is performed in a shared effort within the community applying the central data format.

IV. Selected MDA Applications Making Use of RCE and CPACS

A. The Future Enhanced Aircraft Configurations (FrEACs) Project

The project "Future Enhanced Aircraft Configurations" (FrEACs) is the fourth in a series of DLR projects focused on establishing and utilizing a collaborative product development process for the design of aircraft configurations. The goal of the project is the design of a strut-braced wing as alternative configuration for short- to medium range operations as well as the analysis of the flight mechanical characteristics of a blended-wing-body configuration for long range operations. Especially for the analysis of unconventional configurations - for which none or only a small amount of reference data is available - the utilization of a product development process based on the integration of physical analysis modules is required. Both studies were conducted by combining the expertise of fourteen departments spread across seven different sites within Germany, indicated in Fig. 4. This chapter describes the workflow based design process for the integrated analysis of the strut-braced wing configuration. In this process, RCE is used to interlink the required design competences distributed across the DLR sites.

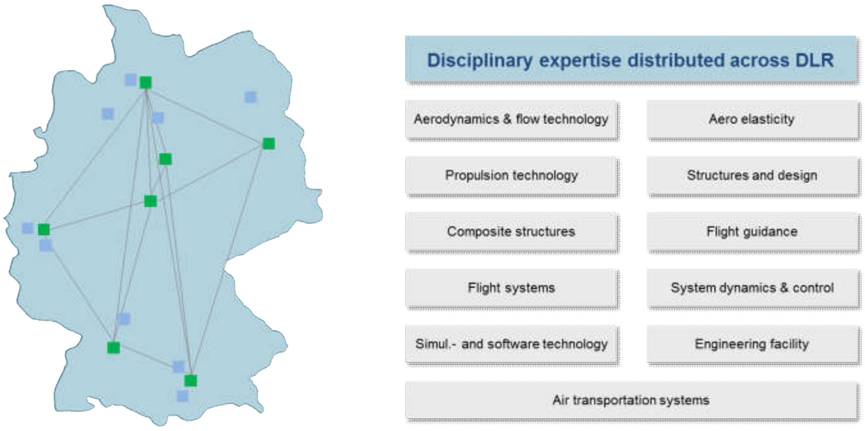

Disciplinary expertise distributed across DLR
Aerodynamics & flow technology
Aero elasticity
Propulsion technology
Structures and design
Composite structures
Flight guidance
Flight systems
System dynamics & control
Simul.- and software technology
Engineering facility
Air transportation systems

Fig. 4. Distribution of disciplinary expertise at different DLR sites across Germany applied within the FrEACs project

The distributed collaborative product development process within the project is based on the availability of a consistent set of engineering services, the application of a common product data model (CPACS), the use of a process integration framework which provides the means for interconnecting the available engineering services within the network of competences, and the application of effective and efficient methods for collaboration among the participants involved in the design process [12].

The engineering services are provided by the different DLR departments and form the important knowledge backbone of the development process. An engineering service is defined as a generically applicable engineering routine, including its automated interface to the collaboration framework and allowing for batch execution. Furthermore, the support of one or more competence specialists owning and sharing the routine as a remote process within the development process is part of the engineering service.

RCE is used to interlink the engineering services hosted across the multiple DLR sites in logical multidisciplinary simulation workflows. Fig. 5 shows a schematic sketch of the server-relay-client network implementation of RCE within the project. A central relay instance of RCE (located in the middle of the figure) provides the basic access and connection point for all the servers and clients within the network. The client instance on the middle-left of the figure connects to the relay instance and is used to create and launch simulation workflows incorporating the available engineering services within the network via RCE's graphical user interface. Furthermore, it can access the result data located on the relay or on server instances within the network. This is also possible via the upper-left and lower-left client instances in the figure, which represent the instances of competence specialists which also have direct access to their own dedicated computation servers. As explained in Chapter II, this allows every involved participant to retrieve the necessary data to debug occurring problems or to inspect results.

To organize the flow of data between the involved engineering services, RCE's components for the management and splitting/merging of XML-based data sets are extensively used in the workflows. Furthermore, the workspace provided in RCE enabled interaction among the involved engineers as well as the structured organization of workflow data.

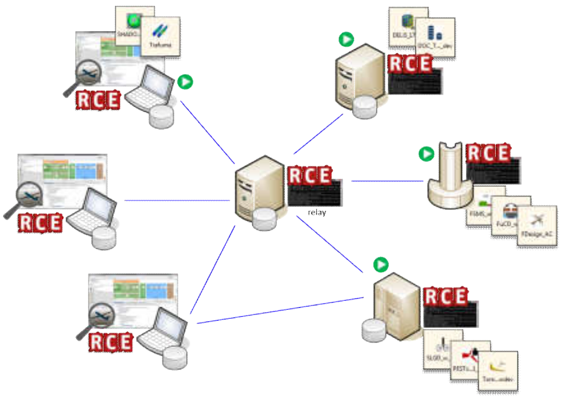


Fig. 5. Server-Relay-Client network implementation using RCE. The relay instance provides the basic access and connection point for all clients and servers within the network.

Enabling collaboration among the participants proved to be a no trivial task, since it requires combining the knowledge of heterogeneous specialists. Within the FrEACs project, “design camps” proved very useful in this matter. A design camp is a special form of project meeting where all participants gather for two and a half days to conduct a special task, e.g., the conceptual design of a configuration. These design camps build upon the results of simulation workflows constructed and executed in RCE. This is enabled by the connectivity and client-server setup of the network established by RCE: every participant involved in the process has access to the simulation data and can provide his/her interpretation on the results.

In FrEACs, the available engineering services were enhanced to cope with the physics of the strut-braced wing configuration. In particular, the introduction of the strut led to the creation and enhancement of engineering services for determining the structural properties of the configuration. The simulation workflow depicted in Fig. 6 was used to test the interconnection of the involved engineering services and to get familiar with design-driving aspects of the configuration. To ensure the correctness of discipline-specific as well as overall results of this workflow, the design camps provided a platform for the required interactions between the involved specialists.

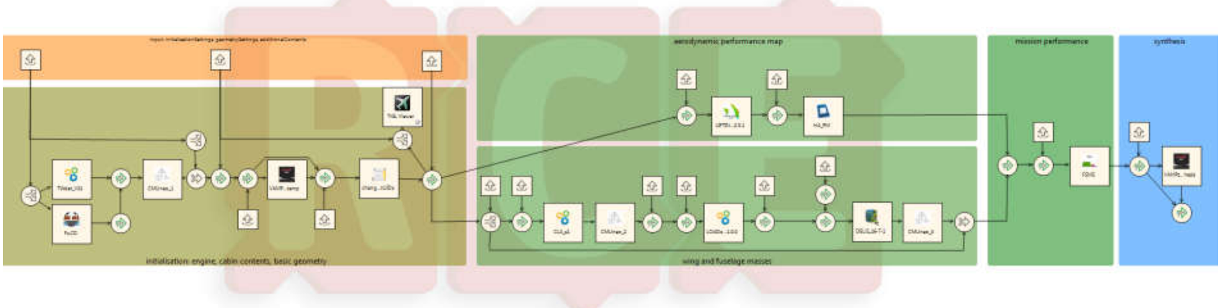

initialization: engine, cabin contents, basic geometry
TiGL Viewer
aerodynamic performance map
wing and fuselage masses
mission performance
synthesis

Fig. 6. The multidisciplinary analysis (MDA) simulation workflow for testing the enhanced strut-braced wing analysis capabilities within process integration software RCE.

Using the knowledge obtained from the development and usage of this linear analysis workflow, a more complex simulation workflow to perform the actual multidisciplinary analysis (MDA) has been set-up, represented in Fig. 7. For this, all identified input-output dependencies between the services were listed and ordered in a large design structure matrix (DSM) (see [3]). Within the workflow, the available engineering services are grouped according to their fidelity level. For each fidelity level considered, a converged and consistent analysis result was obtained. The fidelity levels were loosely coupled within the overall simulation workflow. Since RCE allows for the inspection of intermediate results during the execution of the workflow, this allowed the design team to analyze results after completion of the iterations within a fidelity level, while the higher fidelity calculations have already started. All analysis phases from initialization until analyses at the highest involved level of detail are automatically triggered and base on the results from previous phases. A multi-disciplinary feasible (MDF) approach is chosen for the formulation of the MDA.

The process can be divided into an initialization and subsequent stepwise increasing fidelity level MDF mass iteration parts. In the first part; the Top Level Aircraft Requirements (*TLAR’s)* are set, a cabin layout is generated and mass properties as well as the engine performance map of the involved counter-rotating open-rotor engine is imported. Within the following *initialization* part of the workflow, the geometrical layout of the wings, struts and empennage is created using a combination of empirical correlations and surrogate models. In the subsequent *analyses of level-1 fidelity*; the aerodynamic properties of the configuration and structural mass of the wings, struts, empennage and pylons are determined. With the engine performance and aerodynamic properties known and after updating the mass breakdown of the configuration, a mission simulation is performed to determine the required fuel for the defined reference mission. The physics-based results of the engineering services are then provided to the geometrical layout tool to perform a synthesis on the empennage layout and remaining masses. The analyses within the current level of fidelity are iterated until the operating empty mass and maximum takeoff mass converge.

Afterwards, *analyses of level-2 fidelity* are started which use the converged results as basis. These follow the same MDF convergence procedure as in the level-1 part of the workflow; however incorporate a larger set of load cases and full finite element modelling (FEM) capabilities for the detailed determination of the structural masses of the wings, struts and empennage. Finally, in the *post processing* block, detailed payload-range capabilities are determined and a global FEM model including fuselage and wings is created and structurally sized. All results from the analyses are provided to the team using the RCE workspace based upon which the suitability of the chosen parameter values is assessed during collaborative result interpretation sessions within the organized design camps.

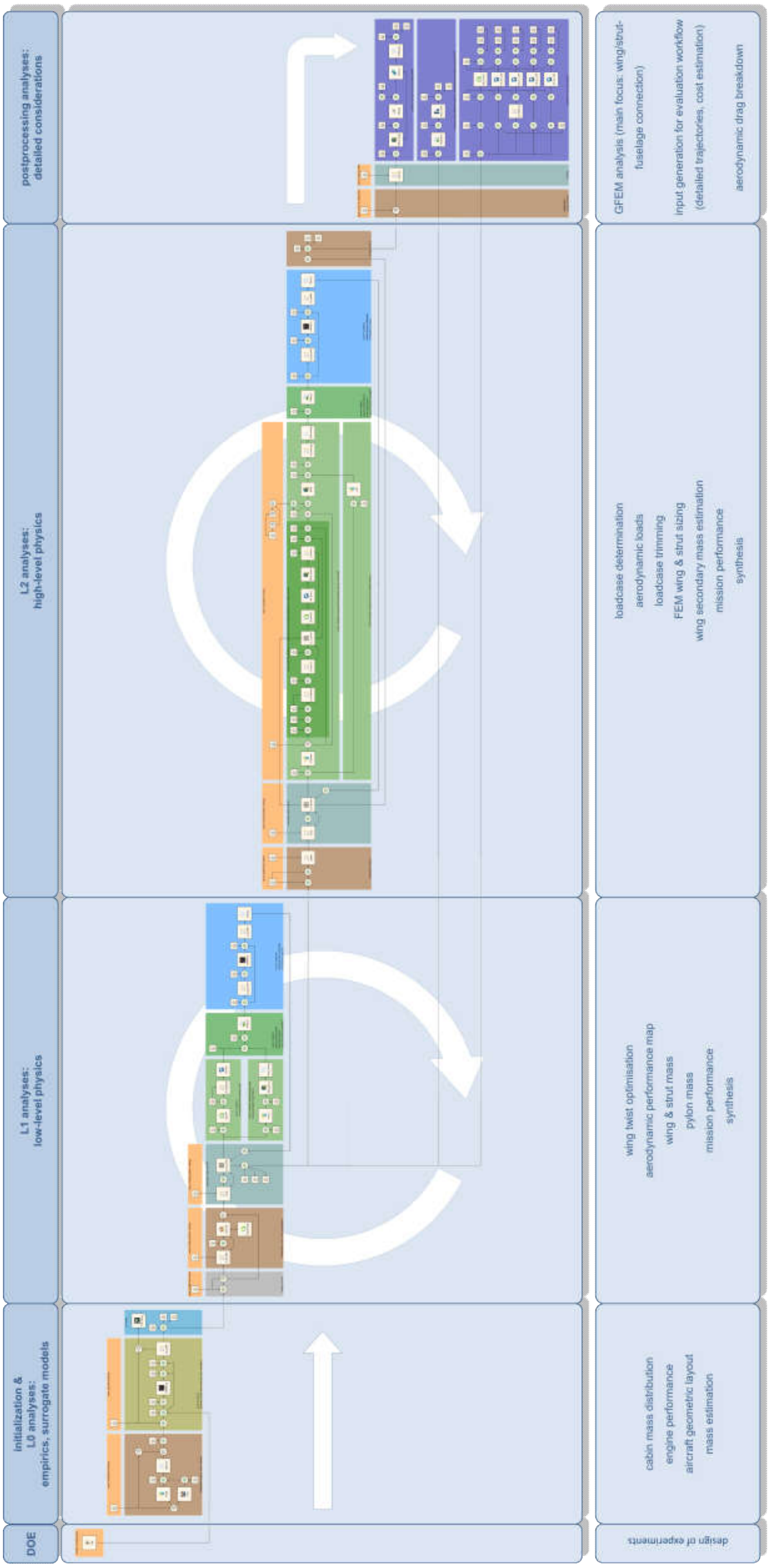
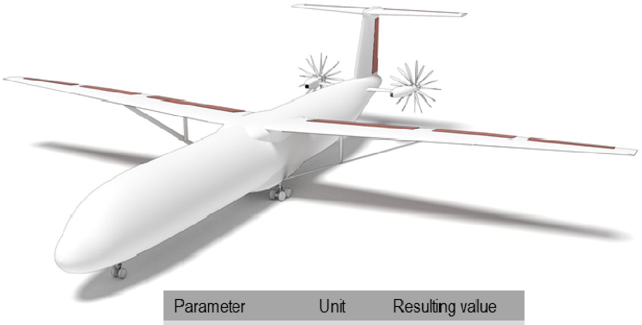


Fig. 7. Simulation workflow for the strut-braced wing, resembling a multi-level approach. The four large groups of engineering services represent the initialization & consecutive analyses on increasing level of detail

Using the workflow, several designs of experiments (DoEs) have been conducted. Fig. 8 shows the parameter variations performed during the strut-braced wing analysis and illustrates the resulting strut-braced wing configuration. Further details are published in [3].

Parameter	Unit	Range
Main wing		
wing aspect ratio	-	15 - 20
wing $t/c_{average}$	-	0.08 – 0.12
wing loading	kg/m²	460 – 540
Strut		
strut η_{wing}	-	0.25 – 0.75
strut depth (* c_{wing})	-	0.2 – 0.5
strut $t/c_{average}$	-	0.08 – 0.14
strut $\#_{juries}$	-	0, 1, 2
Operational		
initial cruise altitude	km	11 - 13
flow laminarity (wing upper side & strut)	-	0% , 50%



Parameter	Unit	Resulting value
Geometry		
strut η_{wing}	-	**0.52**
strut depth	-	**0.15**
# juries	-	**1 (2)**

Fig. 8. Overview of parameter variations performed for the strut-braced wing configuration and impression of the obtained result

B. MEPHISTO Project: Multidisciplinary design studies of agile and highly swept flying wing configurations

The design of agile and highly swept flying wing UCAV (Unmanned Combat Aerial Vehicle) configurations is a challenging task: The assessment of performance and maneuverability requirements, which are essential design drivers for the overall concept, demands for rather detailed data concerning mass distribution, engine performance, and aerodynamic characteristics for a wide range of flight conditions. We investigated such a design case within the DLR research project Mephisto. Its aim was to redesign the so-called SACCON (Stability And Control CONfiguration) UCAV configuration, known from its predecessor projects UCAV-2010 and FaUSST [13], with respect to a set of extended, agility-related design requirements. SACCON is a tailless, lambda-shaped flying wing UCAV concept, characterized by a 53° swept wing with parallel edges for low radar signature purposes. For redesigned concept, the name MULDICON (MULti-DIsciplinary CONfiguration) was chosen.

We designed the MULDICON configuration in close cooperation with the NATO STO task group AVT-251 on “Multi-Disciplinary design and performance assessment of effective, agile NATO Air Vehicles” [14]. It was a rather special design task, as the outer shape of MULDICON (a slightly modified SACCON shape) was already fixed in the beginning and the design task was purely to integrate an aircraft into that shape.

Based on CPACS, RCE, and selected analysis modules we created a workflow for investigating and sizing MULDICON which we show in Fig. 9. It consists of three consecutive steps which are marked in red, blue, and green. Each of the big boxes in the diagram represents one of the distributed analysis tools which have been selected for this process. The data passed among the tools typically consists of a complete CPACS dataset, however, it can also be a set of one or more single parameters which is handed over. The small boxes and circles represent the built-in components of RCE, most prominently the input reader (upwards arrow), which reads and provides additional data to the input, as well as the XML merger (gray arrow with green outline), which joins together data coming from two sources. The combination of a reader module with an XML-Merger is typically used to infuse additional data into the dataset. Such a combination can be seen prior to each of the analysis tools, where tool-specific control data are provided. The remaining built-in components save the current state of the CPACS dataset to a storage medium (downwards arrow) and control the flow of data through the workflow.

The workflow starts with two reader modules in the upper left corner of the diagram. The topmost one reads the initial CPACS dataset and passes it to a TiGL Viewer [11] component, which is a built-in component of RCE that displays the geometry. The leftmost reader component provides a control parameter, specifying where to start the workflow (red, blue, or green step). The other built-in components outside the marked steps are responsible for passing the initial CPACS dataset to the beginning of the selected step.

The second branch (right) creates a set of aerodynamic performance maps: The first map contains force and moment coefficients over a variety of Mach numbers, Reynolds numbers, angles of sideslip, and angles of attack for the "clean" configuration without control surface deflections. On top of this four-dimensional clean configuration dataset, a five-dimensional delta-coefficient performance map is created for each control surface (introducing the deflection of the control surface as fifth dimension). By superposition of different control surface delta-coefficients with the absolute coefficients of the clean configuration dataset it is possible to combine the deflections of multiple control surfaces. A third set of aerodynamic performance maps contains the 18 damping derivatives (6 coefficients x 3 rotation rates) for each point of the clean configuration dataset. Depending on the number of each of the dimensions' entries and on the number of control surfaces, this aerodynamic dataset may grow quite large. In fact, for the MULDICON dataset used here, it accounts for 45,500 entries in total. Even with modern computer systems it is not possible to handle such a number of Reynolds-averaged Navier-Stokes (RANS)-CFD computations in an acceptable timeframe - but using simple, potential flow theory based aerodynamics methods, a performance deck with this size can be created within a few hours or even within minutes. In this process chain, it can be selected whether DLR's open source "LIFTING_LINE" method [15,16, 17] shall be used, or Analytical Methods' commercial "VSAERO" tool [18]. In combination with DLR's simple "HandbookAero" method which accounts for viscous drag and wave drag, a rather complete aerodynamic dataset can be created. Control surfaces like spoilers, which cannot be modeled with inviscid, potential flow based methods, can either be applied via HandbookAero using guessed control derivatives from handbook formulas, or they can be taken from higher-fidelity methods and integrated here manually. At the end of the red block, the resulting performance maps for propulsion and aerodynamics are merged together and the complete CPACS dataset is written to the selected storage medium.

Thereafter, the workflow directly continues with the second step ("Convergence loop", marked in blue), which is the convergence loop for aircraft sizing. It starts with three blocks, integrating template data structures for mass breakdown and weight and balance, as well as starting parameters for the convergence loop. The component labelled "Convergence" is the driver component for an

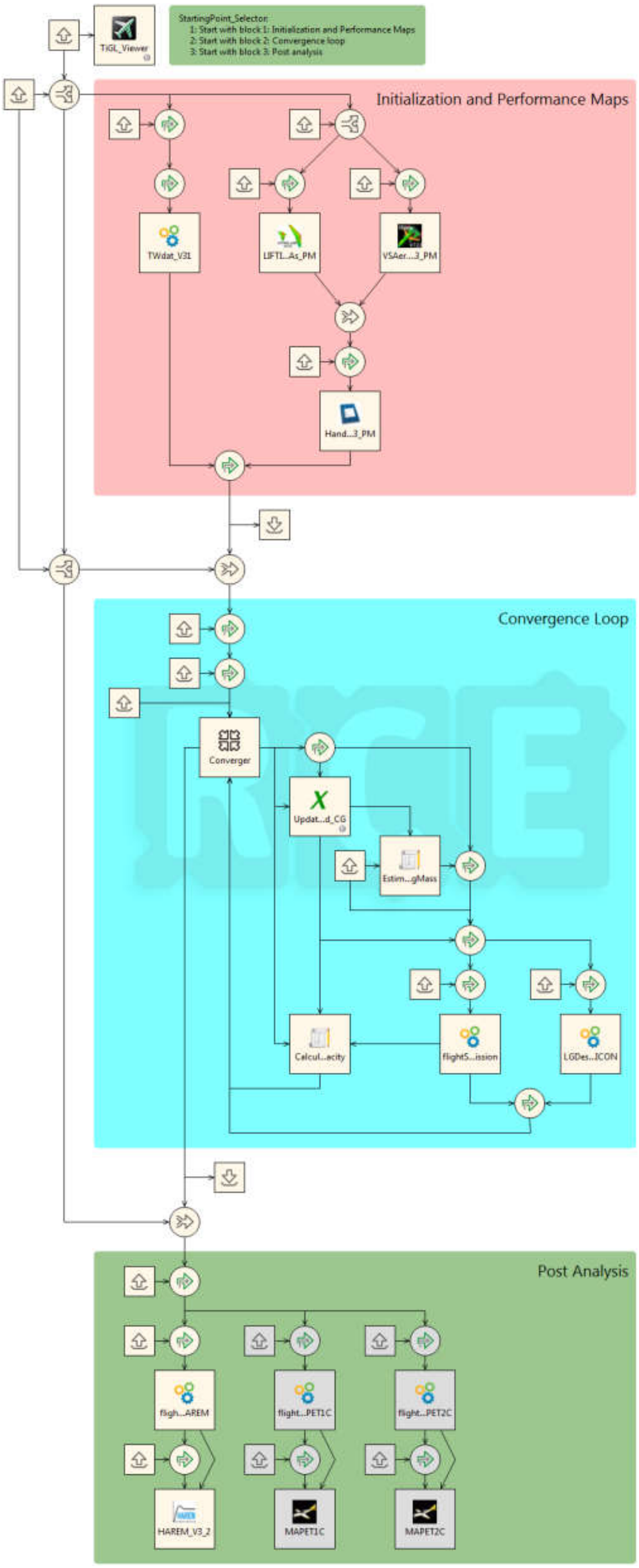


Fig. 9. MULDICON conceptual design workflow.

iteration loop in the workflow. It repeats the rest of the blue part until the computations of design mission fuel and landing gear mass have converged. The iteration loop, starts with a Microsoft Excel spreadsheet (big square with green "X") for calculating mass, CG (**C**enter of **G**ravity) location, and mass moments of inertia for eleven selected weight and balance cases. This is followed by a small Python-script which is responsible for deriving a maximum landing mass for sizing the landing gear. Currently, this is just done by taking a constant factor of 0.9 from the MTOM (Maximum Take Off Mass). Then, the outputs from the Converger, the Excel spreadsheet and the landing mass script are joined together in another XML-Merger component before the workflow splits up into two parallel branches. The left one runs the DLR "flightSim" tool, a 3/6 degree of freedom flight simulation tool [19], which is used here in the 3 degree of freedom mode to perform a simulation of the design mission. As a result, the flight trajectory and the required fuel are written back to the CPACS file. In the right branch the DLR tool "LGDesign" is used to analyze and size the landing gear of MULDICON [20]. As a result, it provides mass and geometry of the landing gear, which is combined with the mission fuel mass and trajectory from flightSim and fed back into to Converger module for the next iteration. The "CalculateRequiredFuelCapacity" component (lower left corner of the blue region) is just a simple Python-script which computes a required fuel capacity in percent based on the mission fuel demand resulting from flightSim. At the end of the blue block, the aircraft is completely sized and the results are again written to a CPACS file.

The third step ("Post analysis", marked in green) consists of different analysis tools, further investigating the properties of the aircraft. On the left side of the workflow, there is "HAREM", a DLR analysis tool for investigating and evaluating the handling qualities of an aircraft [21,22,23]. In this branch, the flightSim tool is used again, but this time it creates a 6 degree of freedom dynamic aircraft model which is handed over to HAREM. HAREM is used to assess some of the design requirements like the roll performance. The other two branches are intended for future flight performance and landing performance investigations, however, they are not used yet.

Using this RCE workflow it was possible to investigate the MULDICON design and size and position the main components. As typical for flying wing configurations, a special focus had to be laid on the location and movement of the CG. To give an impression of the very tight limitations here, it shall be mentioned that for the operational empty mass case, the CG movement due to retracting the landing gear already uses around one third of the permitted CG range. Finally, a good compromise for the internal arrangement was found. In order to go beyond current possibilities to visualize a CPACS dataset using the TiGL Viewer, the Microsoft Excel spreadsheet from the MULDICON workflow was also used as a construction table for the Dassault CATIA CAD software, resulting in the 3D visualization shown in Fig. 10. Further details regarding the MULDICON design work are published in [24] and [25].

Even though the RCE workflow described above was used as a pure MDA application within Mephisto, it could easily be encapsulated by some driver component to perform trade studies, DoE and optimization runs for further enhancement.

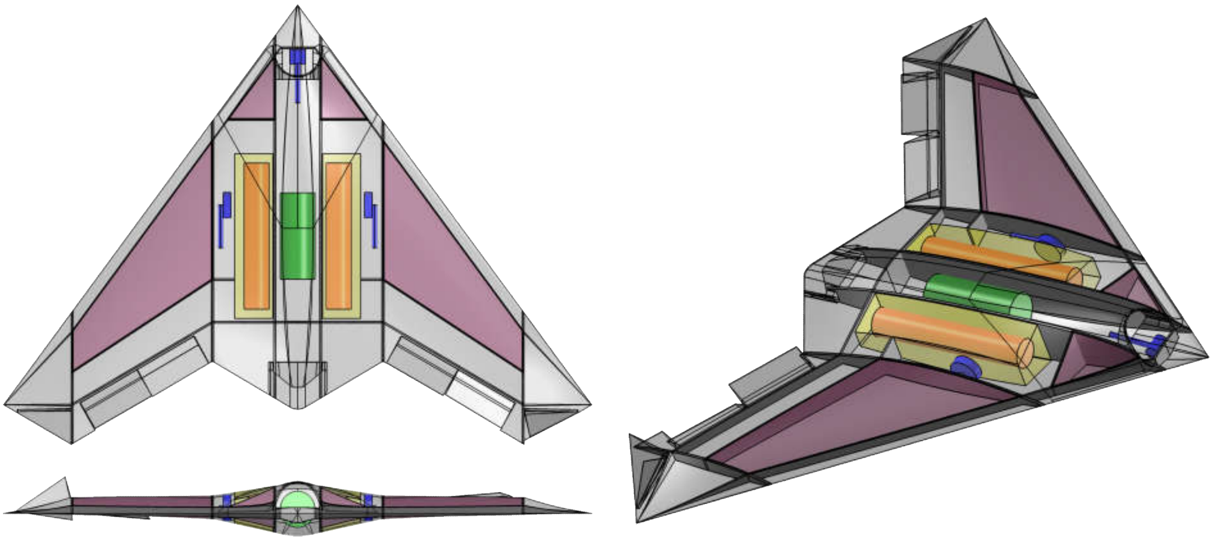
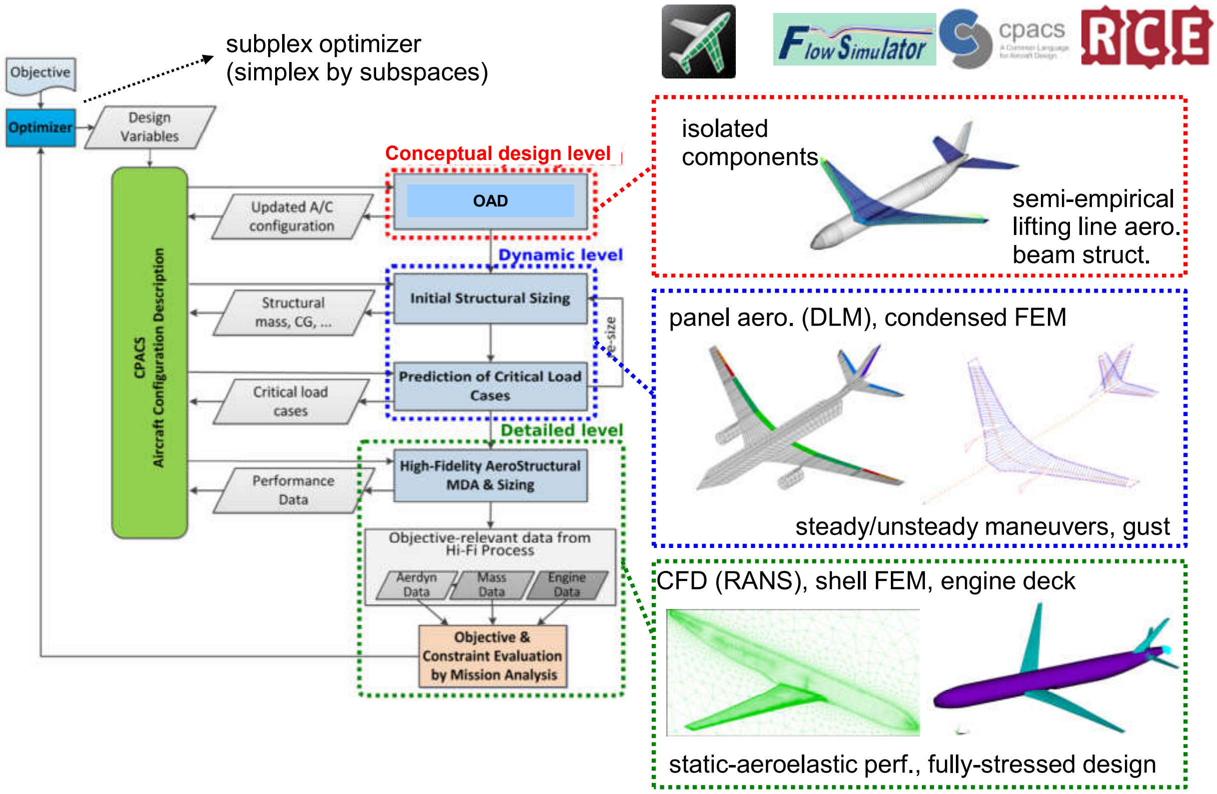


Fig. 10. 3D model of MULDICON with internal arrangement.

V. Selected MDO Applications Making Use of RCE and CPACS

A. Digital-X Project: Collaborative Multi-Level MDO Process Development and Application

Within the DLR project Digital-X [26] a new challenge for RCE was to support the use of high-fidelity analysis tools, some of which were to run on HPC resources. Up to that point, RCE was applied mostly to assemble processes with lower-fidelity tools, such as in conceptual aircraft or spacecraft design. Moreover, the projected MDO process in Digital-X was to be a multi-level one, comprising tools and sub-processes ranging from low-fidelity, over mid-fidelity, to high-fidelity and computation-intensive [27,28,29]. It involved disciplinary expert groups from eight DLR institutes, each committed to integrating their tools and sub-process into the overall MDO process. A schematic view of the target process is shown in Fig. 11.



subplex optimizer
(simplex by subspaces)
Objective
Optimizer
Design Variables
CPACS
Aircraft Configuration Description
Conceptual design level
OAD
Updated A/C configuration
Dynamic level
Initial Structural Sizing
Structural mass, CG, ...
re-size
Prediction of Critical Load Cases
Critical load cases
Detailed level
High-Fidelity AeroStructural MDA & Sizing
Performance Data
Objective-relevant data from Hi-Fi Process
Aerdyn Data
Mass Data
Engine Data
Objective & Constraint Evaluation by Mission Analysis
Flow Simulator
cpacs
RCE
isolated components
semi-empirical
lifting line aero.
beam struct.
panel aero. (DLM), condensed FEM
steady/unsteady maneuvers, gust
CFD (RANS), shell FEM, engine deck
static-aeroelastic perf., fully-stressed design

Fig. 11. Multi-level/multi-fidelity MDO process in DLR's Digital-X project (schematic) & disciplinary models

To support these new requirements, RCE was extended in several ways. The most obvious extension was implementing a built-in component to support HPC clusters. This component sends data to a cluster, submits the job to the cluster's queuing system, waits for it to start or finish, fetches the results and propagates them into the workflow. Other adaptations included enhancements to workflow engine, network and storage layers to handle very large data sets more efficiently (on the order of terabytes (Tb) per iteration). The process sketched in Fig. 11 was implemented in RCE as shown in Fig. 12.

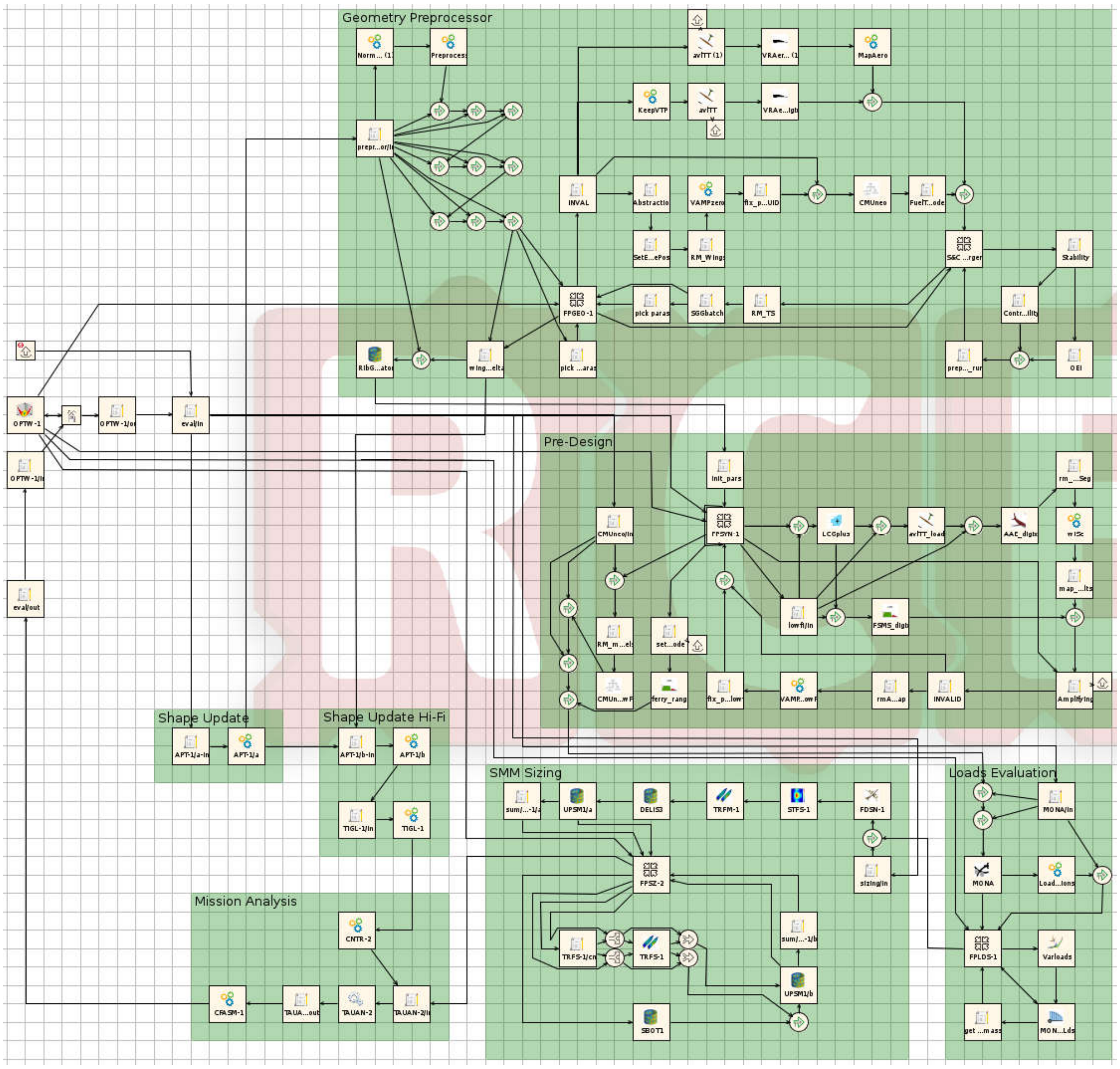


Fig. 12. RCE implementation of the Digital-X MDO process

The components (disciplinary tools) in a running workflow were executed on workstations at six DLR institutes and on one HPC cluster which is also part of the DLR network. Both the RCE tool nodes and the workflow controller node connected to an instance configured as a relay server (in fact, to two of them, for a fail-over backup), in order to make all tools in the network visible to the controller node. The operating system differed between the tool nodes. This hardware and network environment is depicted in Fig. 13.

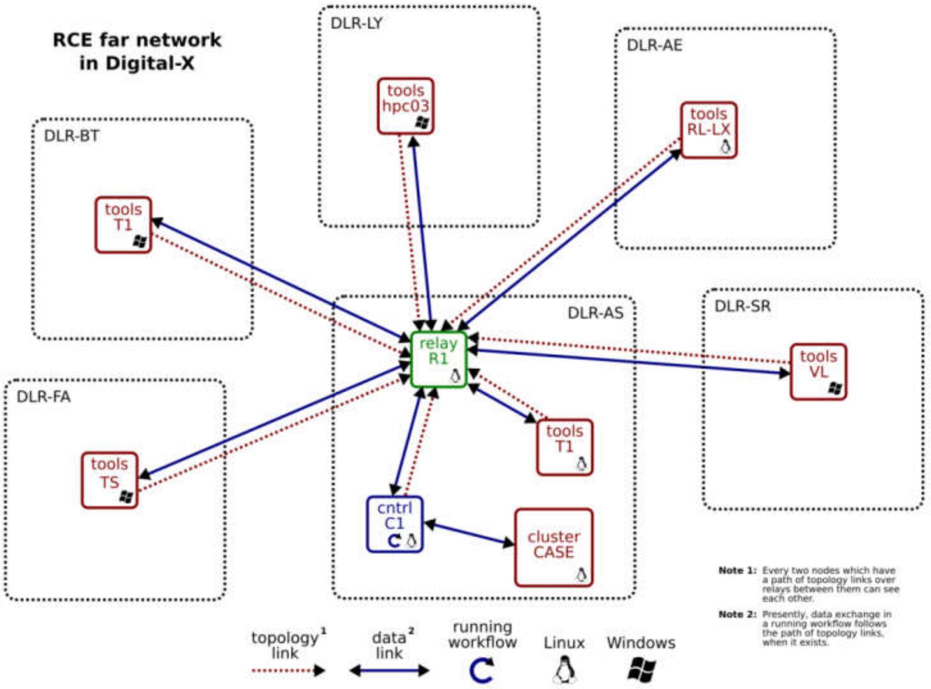

RCE far network in Digital-X
DLR-LY
tools hpc03
DLR-AE
tools RL-LX
DLR-BT
tools T1
DLR-AS
relay R1
DLR-SR
tools VL
tools T1
DLR-FA
tools TS
cntrl C1
cluster CASE
Note 1: Every two nodes which have a path of topology links over relays between them can see each other.
Note 2: Presently, data exchange in a running workflow follows the path of topology links, when it exists.
topology¹ link
data² link
running workflow
Linux
Windows

Fig. 13. Network of tool nodes and workflow controller for running the Digital-X MDO workflow: each black-dotted box represents a DLR institute at a particular DLR site in Germany, red boxes represent tool nodes or HPC clusters, the blue box represents the workflow controller, the green box is the relay node used to make tool nodes visible to the workflow controller and receives and forwards data during the execution of the process

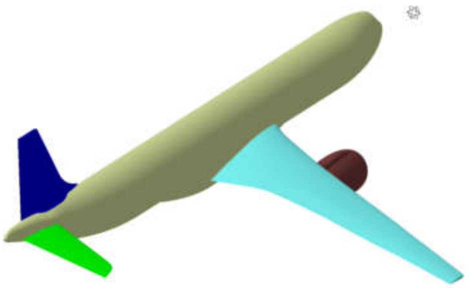


Fig. 14. Half-model of generic long-range transport aircraft used as a baseline for MDO in Digital-X

The high-fidelity aspect of the workflow introduced very long run times as well. Table 1 shows the run time by sub-process (composed of one or more tools), which summed to the total run time of about 24 hours. However, when executing the workflow, we were not yet able to use the recently released version of RCE which includes the robustness features detailed above in Chapter II. Hence, the workflow had to be restarted often from a state saved at each outer optimizer iteration, due to network failures, tool node restarts (e.g., after installing security updates), and tool node overload (e.g. no more disk space left). This brought the “effective” run time of a single design analysis, computed as the total workflow run time divided by the number of evaluated designs, to about 56 hours.

Table 1. Runtime breakdown of a single design analysis

Sub-process	Run time [h]	Of total [%]
Preliminary design	1.3	5.5
Loads	5.8	24.6
Structure sizing	15.2	64.4
Mission (Breguet)	1.3	5.5
Total	**23.6**	**100.0**

The MDO problem was to minimize mission fuel burn of a twin-engine long-range airliner. The baseline aircraft configuration is shown in Fig. 14. All design constraints (such as stability margin, landing/takeoff distance, or structure failure criteria) were satisfied within the respective disciplinary sub-processes, so that the outer optimizer worked on an unconstrained optimization problem. Although the complete aircraft configuration was analyzed, the design parametrization was for the wing only: Table 2 shows the list of nine selected design parameters, consisting of seven planform parameters and two section parameters, their limits for the optimization, the baseline and optimized values. The outer optimizer was the derivative-free Subplex method. More details on the setup of the MDO process, the disciplinary models and the optimization results can be found in [28,29].

Table 2. Selected design parameters

Parameter	Min.	Max.	Base	Opt.	Δ
Aspect ratio (AR)	7	12	9.2	9.38	+0.18
Sweep (φ_{LE}) [°]	24	40	32	32.9	+0.9
Taper ratio 2-3 (n_{23})	0.20	0.80	0.59	0.572	-0.018
Taper ratio 3-4 (n_{34})	0.20	0.80	0.55	0.532	-0.018
Twist 2 (ε_2) [°]	-6	6	0.5	0.14	-0.36
Twist 3 (ε_3) [°]	-6	6	1.0	0.64	-0.36
Twist 4 (ε_4) [°]	-6	6	-2.0	-1.64	+0.36
Section 2 thickness change (δt_2) [%]	-25	25	0	-10.2	-10.2
Section 4 thickness change (δt_4) [%]	-25	25	0	1.2	+1.2

Given the long effective run time of a single design analysis of about 56 hours, at the conclusion of the project the optimization was not yet fully converged. However, we could confirm the general feasibility of highly collaborative, multi-level MDO with many disciplinary groups. Regarding the workflow run time and causes of interruptions, in the time since RCE was improved to handle more robustly network failures. Solving the issues stemming from tool nodes lacking uptime, computational and storage resources, however, lies outside the scope of RCE in its role as an integration framework. This motivated moving to an all-HPC based solution, called Cybermatrix MDO protocol, in the follow-on DLR project VicToria [30]. The Cybermatrix MDO protocol is a novel approach towards computationally demanding and collaboration intensive MDO and its implementation is described in detail in [31].

B. EU-Project AGILE: Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts

Within the EU-funded AGILE project [32], the focus was on streamlining and accelerating the deployment of large-scale design and optimization processes. The first challenge one was to drastically reduce the time needed to implement an MDO process within a process integration design and optimization environment. For complex products the input and output (I\O) parameters which are handled and exchanged by models and tools might reach the order of several thousands. Up to this point, most of the MDO processes require time consuming activities in order to manually connect all the tools involved, until the desired MDO strategy is implemented. Even when standards are adopted to facilitate the exchange of data (such as CPACS), the number of tools composing the MDO problem might quickly grow to the point to inhibit desired changes to the MDO process under deployment. The second challenge is to enable the composition of MDO processes which integrate multiple engineering services (such as a specific disciplinary simulation) which are provided and hosted by different organizations. Interoperability of platforms and integration of existing legacy processes as sub-processes are key enablers in such scenarios.

In AGILE a total of 19 partners from industry, research and academia from Europe, Canada and Russia have developed multiple solutions to cope with the challenges of collaborative design and optimization of complex products. During the project multiple extensions have been developed for RCE in order to:

1. Enable the automatic implementation of workflows.
2. Support the cross-organizational execution of workflows.

1. Automatic implementation of workflows

Any MDO process can be de-facto described by an XDSM (eXtended Design Structure Matrix) [33] representation. Especially during the architecting phase of an MDO process, such a representation provides a way to describe multiple MDO strategies. Within AGILE a machine interpretable MDO schema (CMDOWS) containing all the elements available within an XDSM representation has been developed. As companion to the schema definition, a dedicated RCE parser has been developed, which materializes RCE workflows from the MDO schema definition. This enabled the design teams with the flexibility to change or adapt the MDO strategy by manipulating the XDMS schema, without the need to manually re-implement the RCE the workflows, which are instead updated automatically.

We provide an example for a simple problem in Fig. 15: here, two MDO architectures are illustrated as XDSM for the Sellar problem. These two architectures are at first saved into the MDO model CMDOWS, and in a second step the RCE workflows are automatically generated and can then be directly imported into RCE.

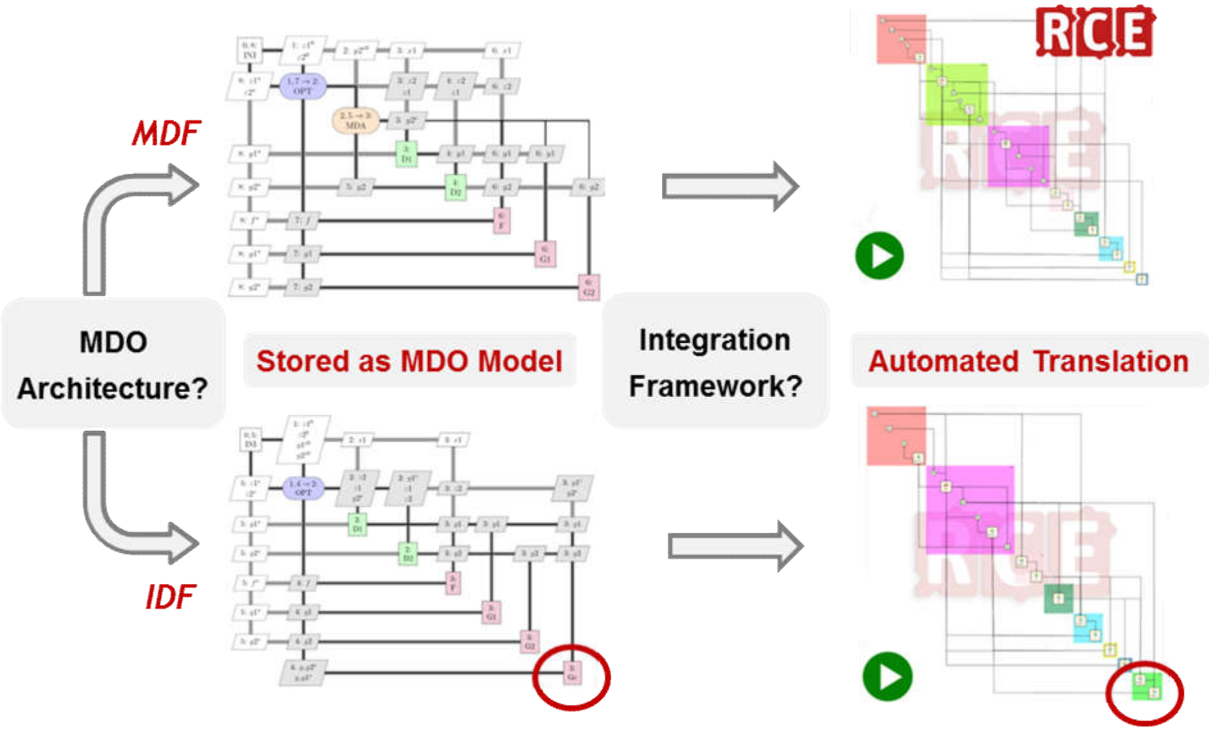
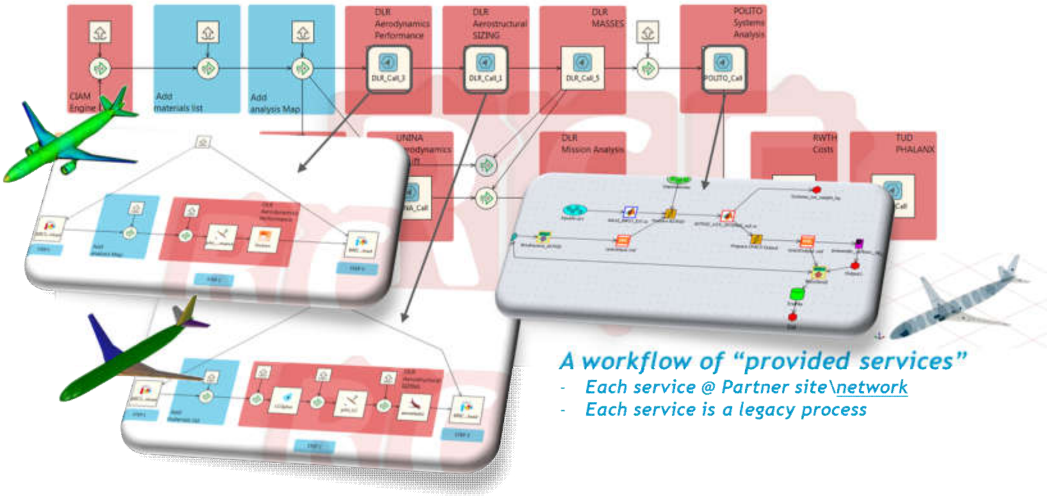


Fig. 15. The Sellar problem: MDO formulation as XDSM, saved as MDO models in CMDOWS format, and automatically generated as RCE workflows implementation.

2. Cross-organizational execution of workflows

An additional proprietary component called “remote service component” was developed for RCE to enable the implementation of “workflows of engineering services” which are distributed between multiple organizations. To this purpose such RCE component serves as a proxy for a generic engineering service which is hosted in a different company and eventually implemented within a different MDO platform. Therefore, an MDO master workflow is generated (automatically) which consists only of RCE “remote service” components. At execution time, RCE will take care of executing the MDO workflow implemented, as it would do for a standard RCE workflow. In addition the “remote service” components take care to forward the data to be analyzed to the remote organizations, and fetch the results back when available. The communication between different organization networks is taken care of by a collaborative architecture developed during the project [34].

We illustrate an example for a “workflow of engineering services” in Fig. 16, in which the master MDO workflow is implemented in RCE via the “remote service” components, which provide access to legacy sub-workflows hosted at different organizations.



A workflow of "provided services"

- Each service @ Partner site\network
- Each service is a legacy process

Fig. 16. Service Oriented Architecture: Master RCE workflow composed by "remote service" components, working as proxy for sub-workflows hosted at different organizations and implemented in heterogeneous platforms (RCE or others).

Within the AGILE project several conventional and unconventional aircraft use cases have been investigated by making use of the mentioned technologies. We show an example for a strut-brace wing aircraft in Fig. 17 [35]. Here, the formulated MDO process (an MDF architecture) is automatically implemented in a RCE workflow, composed by remote services hosted at different organizations.

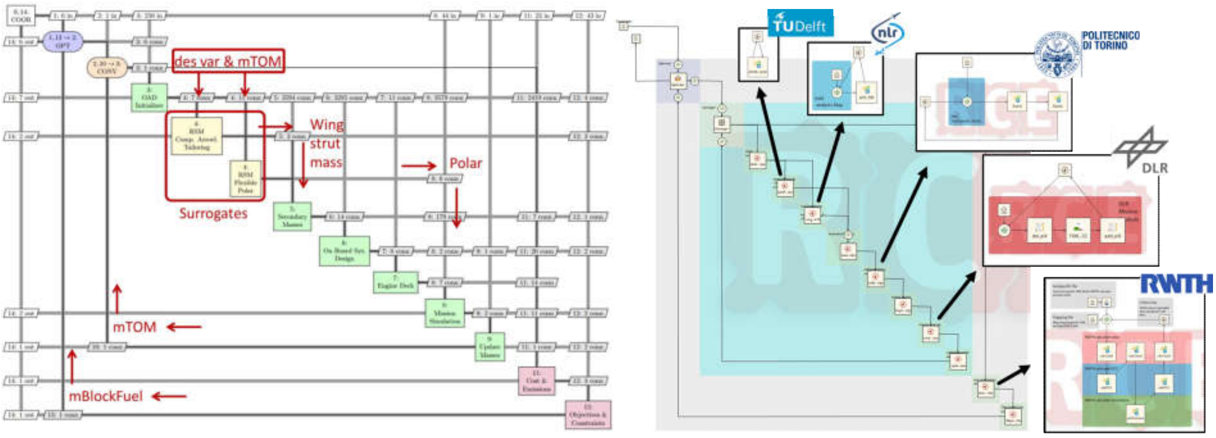


Fig. 17. AGILE strut-braced-wing use case. Left: XDSM representation of the MDO process. Right: corresponding implemented RCE workflow, automatically generated as "workflow of engineering services".

C. VicToria Project: Integrated Process Chain for Aerostructural Wing Optimization Using a Distributed Analysis Framework

In the DLR project VicToria three different MDO approaches are pursued [36]. Two of these approaches use a Python framework for running all simulation programs and tools in an HPC environment [37]. In contrast to these two approaches, the integrated process chain for aerostructural wing optimization presented herein follows a distributed analysis approach. This MDO process chain is based on high-fidelity simulation methods and is a further development of the process chain that was developed in the collaborative research project AeroStruct and funded by the German National Aeronautics Research Program (LuFo IV) [38,39,40]. We use this process chain to better understand multidisciplinary interactions in aerostructural optimization of flexible wings and to assess load alleviation technologies.

The employed disciplinary analysis and simulation tools are making use of typical aerospace industry standards (CAD modeling based on CATIA® V5, CFD and FSI based on the FlowSimulator [41,42] environment, structural analysis using MSC Nastran, structural sizing using HyperSizer, mesh generation using Pointwise). The challenge was to integrate commercial software into the process chain, including software that only runs under Windows on desktop computers and other software that is executed on a Linux HPC cluster. This integration has been successfully realized with the help of RCE. The main RCE tasks here are file transfer and running applications in a distributed framework with different operating systems. Due to the use of RCE, a robust and secure solution with runtimes on the order of weeks is available. Furthermore, the control of RCE workflows with the supported SSH interface allows the efficient execution in automated HPC centered processes.

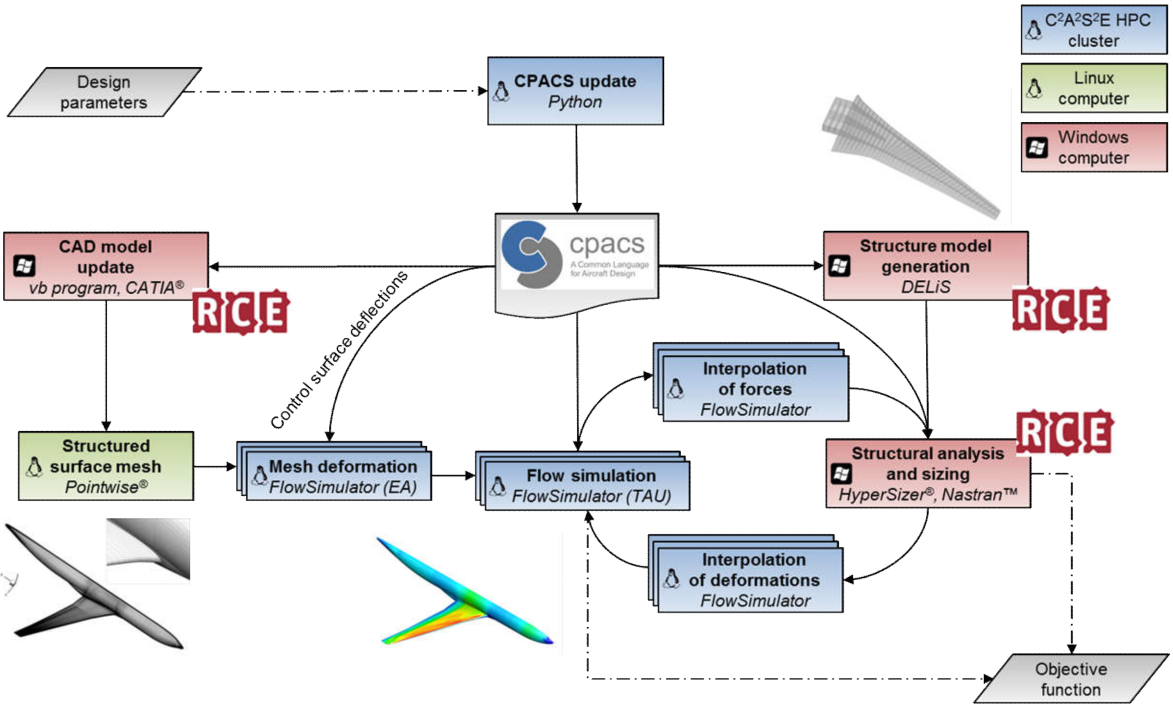


Fig. 18. VicToria project: integrated process chain for aerostructural wing optimization based on high-fidelity simulation methods using RCE to integrate commercial Windows programs and distributed analysis processes in an HPC environment.

Fig. 18 shows the integrated process chain for aerostructural wing optimization using high fidelity simulation methods. The main characteristics of the developed process chain are as follows:

- Usage of a central file format for parametric aircraft description (CPACS),
- Automated grid deformation for aerodynamic simulations,
- Automated structural model generation for structural simulations,
- Parallel static aeroelastic analysis for an arbitrary number of flight and load cases,
- Structural wing box sizing for composite structures,
- Applicability for large geometrical changes,
- A global optimization strategy.

The selected MDO architecture falls in the category of MDF-optimization. In the MDF architecture a full multidisciplinary analysis (MDA) is performed each iteration of the optimization. This means that the investigated design fulfills all constraints in each optimization step and hence is called a feasible design.

Usually, the starting point for a wing optimization is a detailed geometrical model of a given reference aircraft configuration. From this nonparametric model a fully parametric description of the aircraft using the Common Parametric Aircraft Configuration Schema (CPACS) has to be generated. All disciplinary simulation programs in the process chain provide interfaces to this central hierarchical and human readable file format. Based upon a design parameter variation and a following transfer to the CPACS dataset the disciplinary models are built or updated automatically. Thereby, the design parameters describe the wing planform including twist and airfoil thickness distributions and the orthotropy direction of the composite structure.

For the CAD model update in CATIA and the structure model generation with DELiS [43] the RCE framework is used. Both programs provide their functionalities as a published component on a RCE tool server. The static aeroelastic analysis is then run in parallel for all flight cases under cruise flight conditions and all maneuver load cases. In the actual implementation, the process chain is limited to steady state maneuver load cases and only the wing fuselage configuration is analysed within the high-fidelity simulation process. For each flight case and each load case the surface pressure distribution and aerodynamic coefficients of the wing are determined by solving the RANS equations within a numerical flow simulation. Elastic characteristics of the wing and its internal loads are determined using the FEM. Subsequently, the wing mass is deduced by processing these internal loads. The interactions between the aerodynamic forces and the structural deformations of the elastic wing are taken into account in the static aeroelastic analysis. Thereby, the fluid-structure coupling loop stops when the values for the lift to-drag ratio, wing mass and fuel consumption are converged.

Within the parallel static aeroelastic analysis the wing box structure is sized and the bending and torsional stiffness of the wing converge in the fluid-structure coupling loop. Thereby, the structural sizing forms an inner loop to fulfill the structural constraints in terms of failure criteria and fulfills the margins of safety and the wing mass for a fixed aerodynamic load. The structural analysis and sizing process [44] is based on the commercial programs HyperSizer and MSC Nastran and is integrated into the process chain by using RCE. The main results of this parallel analysis are the wing mass and the deformed wing shapes for the flight cases under cruise flight conditions, which are normally called “1g-flight shapes”. Based on these 1g-flight shapes the aerodynamic performance in terms of lift-to-drag ratios L/D is determined for the selected flight missions.

The last step in the process chain is the evaluation of the objective function for the multidisciplinary assessment of the wing design. In general the objective function defines a weighted fuel consumption of different flight missions. The optimization algorithm then calculates a new set of values for the design parameters based on the value of the objective function. For the aerostructural wing optimization a surrogate based optimization (SBO) method [45] has been used successfully. This optimization method searches the global optimum and offers a high level of robustness.

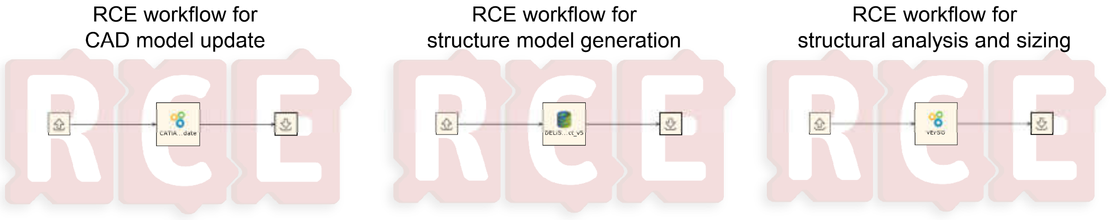


Fig. 19. RCE workflows of the VicToria integrated high-fidelity MDO process chain.

In Fig. 19 the three simple RCE workflows are shown, which are an integral part of the overall process chain. These RCE workflows are running in batch mode and allow the robust and efficient control of the used commercial programs on Windows operating systems.

VI. Conclusions and Outlook

DLR’s remote component environment (RCE) is an open source, distributed integration environment. It supports the integration of custom tools and their composition into automated, executable process chains, so-called simulation workflows. The key features of RCE have been presented and the use of RCE for implementing selected collaborative MDA and MDO workflows in DLR projects has been discussed. Being an open source tool, RCE is being used in many other multidisciplinary design and optimization projects and has become an essential tool for many engineers and scientists at DLR and for external users worldwide. By working in close collaboration with users, we ensure that RCE not only satisfies the specific requirements of the users, but also that it is a versatile framework that transparently supports users in their daily tasks. We are continuously improving RCE for its continuously growing user base. To this end, we are currently pursuing a number of projects, some of which aim to implement specific user requirements, while others serve to strengthen RCE’s foundation and to ensure the continued development of RCE for the years to come.

While RCE allows for the configuration of all features via its graphical user interface, supports users via its context-sensitive online help, and introduces new users via an external user guide, it does not yet feature a complete and consistent usability concept. We are currently working on designing and implementing such a concept to improve the user experience, and further increase user productivity.

Moreover, the central code coordinating the execution of workflows and tools was originally designed for much simpler requirements than those imposed by contemporary workflows. In particular, a mapping of tools to executing machines currently has to be defined before the start of the workflow.This fixed mapping prevents us from dynamically assigning idle machines to the execution of tools, which would lead to improved scalability of large workflows.

Also, users have noted that the maintenance of complicated workflows could be simplified by allowing for the hierarchical inclusion of "sub-workflows". This necessitates, however, adaptations to the central code for execution of workflows in order to support such sub-workflows as virtual tools. We are currently overhauling the central code governing the workflow execution to, among other improvements, increase its performance and maintainability, to support the above-mentioned automatic load-balancing, and to account for sub-workflows.

Finally, we are aiming to make RCE's pre-integrated optimizer component more flexible. This component is currently hardwired to provide the optimization algorithms implemented by the Dakota framework. We are working to enable users to substitute arbitrary MDAO-frameworks for this hardwired one, thus providing them with the possibility to greatly decrease the running time and increase precision of their design and optimization workflows.

References

[1] D. Seider, M. Litz, A. Schreiber, P. M. Fischer and A. Gerndt, "Open source software framework for applications in aeronautics and space," in IEEE Aerospace Conference, Big Sky, Montana, USA, 2012.

[2] D. Seider, A. Basermann, R. Mischke, M. Siggel, A. Tröltzsch and S. Zur, "Ad hoc Collaborative Design with Focus on Iterative Multidisciplinary Process Chain Development applied to Thermal Management of Spacecraft," in 4th CEAS Air & Space Conference, Linköping, Sweden, 2013.

[3] E. Moerland, T. Pfeiffer, D. Böhnke, J. Jepsen, S. Freund, C. M. Liersch, G. Chiozzotto, C. Klein, J. Scherer, Y. J. Hasan, and J. Flink, "On the Design of a Strut-Braced Wing Configuration in a Collaborative Design Environment," in 17th AIAA Aviation Technology, Integration, and Operations Conference, AIAA Paper 2017-4397, Denver, United States of America, 2017.

[4] German Aerospace Center (DLR), Simulation and Software Technology, "Remote component envirnment (RCE)," [Online]. Available: http://www.rcenvironment.de. [Accessed 14 May 2019].

[5] A. Ronzheimer, F. J. Natterer, J. Brezillon, "Aircraft Wing Optimization Using High Fidelity Closely Coupled CFD and CSM Methods," AIAA-Paper 2010-9078. 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference, 2010.

[6] P. Ciampa, B. Nagel, G. La Rocca, "Preliminary Design for Flexible Aircraft in a Collaborative Environment," Proc. 4th CEAS Air & Space Conference, 16-19 Sep 2013, Linköping, Sweden, 2013.

[7] C. Liersch and M. Hepperle, "A distributed toolbox for multidisciplinary preliminary aircraft design," CEAS Aeronautical Journal, vol. 2, no. 1, pp. 57-68, 2011.

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

[9] German Aerospace Center (DLR), Institute of System Architectures in Aeronautics, "Common Parametric Aircraft Configuration Schema (CPACS)," (Online). Available: http://www.cpacs.de. [Accessed 14 May 2019].

[10] German Aerospace Center (DLR), Simulation and Software Technology, "TIXI XML interface," [Online]. Available: https://github.com/DLR-SC/tixi. [Accessed 14 May 2019].

[11] German Aerospace Center (DLR), Simulation and Software Technology, "TIGL Geometry Library," [Online]. Available: https://dlr-sc.github.io/tigl/. [Accessed 14 May 2019].

[12] E. Moerland, R. G. Becker and B. Nagel, "Collaborative understanding of disciplinary correlations using a low-fidelity physics-based aerospace toolkit," CEAS Aeronautical Journal, vol. 6, no. 3, pp. 441-454, 2015.

[13] C. M. Liersch, K. C. Huber, A. Schütte, D. Zimper and M. Siggel, "Multidisciplinary design and aerodynamic assessment of an agile and highly swept aircraft configuration," CEAS Aeronautical Journal, Vol. 7, No. 4, 2016, pp. 677–694, DOI: 10.1007/s13272-016-0213-4.

[14] R. M. Cummings, C. M. Liersch and A. Schütte, "Multi-Disciplinary Design and Performance Assessment of Effective, Agile NATO Air Vehicles," AIAA AVIATION Forum, American Institute of Aeronautics and Astronautics, 2018, DOI: 10.2514/6.2018-2838.

[15] “LIFTING_LINE Homepage,” URL: https://www.dlr.de/as/en/desktopdefault.aspx/tabid-188/379_read-625/, [Accessed 8 May 2019].

[16] K. H. Horstmann, “Ein Mehrfach-Traglinienverfahren und seine Verwendung für Entwurf und Nachrechnung nichtplanarer Flügelanordnungen,” Tech. rep., Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt, 1987.

[17] C. M. Liersch and T. F. Wunderlich, “A Fast Aerodynamic Tool for Preliminary Aircraft Design,” Multidisciplinary Analysis and Optimization Conference, American Institute of Aeronautics and Astronautics, 2008, DOI: 10.2514/6.2008-5901.

[18] “VSAERO Homepage,” URL: https://starkaerospace.com/products-services/ami/software/, [Accessed 8 May 2019].

[19] G. Looye, “TECS/THCS-based generic autopilot control laws for aircraft mission simulation,” Second CEAS Specialist Conference on Guidance, Navigation and Control, 2013.

[20] W. Krüger, S. Cumnuantip and C. Liersch, “Multidisciplinary Conceptual Design of a UCAV Configuration,” RTO/AVT Panel Workshop "Virtual Prototyping of Affordable Military Vehicles Using Advanced MDO", 2011.

[21] J. Schwithal, D. Rohlf, G. Looye and C. M. Liersch, “An innovative route from wind tunnel experiments to flight dynamics analysis for a highly swept flying wing,” CEAS Aeronautical Journal, Vol. 7, No. 4, 2016, pp. 645–662, DOI: 10.1007/s13272-016-0214-3.

[22] G. Duus and H. Duda, “HAREM - HAndling Qualities Research and Evaluation using Matlab,” IEEE International Symposium on Computer Aided Control System Design, Kohala Coast, HI (us), 22-27 August 1999, 1999, pp. 428–432.

[23] J. Ehlers, “Flying Qualities Analysis of CPACS Based Aircraft Models - HAREM V2.0 -,” DLR internal report 111-2013/21, German Aerospace Center (DLR), Institute for Flight Systems, June 2013.

[24] C. M. Liersch and G. Bishop, “Conceptual Design of a 53deg Swept Flying Wing UCAV Configuration,” AIAA AVIATION Forum, American Institute of Aeronautics and Astronautics, 2018, DOI: 10.2514/6.2018-2839.

[25] C. M. Liersch, A. Schütte, M. Siggel and J. Dornwald, “Design Studies and Multi-Disciplinary Assessment of Agile and Highly Swept Flying Wing Configurations”, Deutscher Luft- und Raumfahrtkongress 2018.

[26] N. Kroll, M. Abu-Zurayk, D. Dimitrov, T. Franz, T. Führer, T. Gerhold, S. Görtz, R. Heinrich, Č. Ilić, J. Jepsen, J. Jägersküpper, M. Kruse, A. Krumbein, S. Langer, D. Liu, R. Liepelt, L. Reimer, M. Ritter, A. Schwöppe, J. Scherer, F. Spiering, R. Thormann, V. Togiti, D. Vollmer, J.-H. Wendisch, „DLR Project Digital-X: Towards Virtual Aircraft Design and Flight Testing based on High-Fidelity Methods,“ CEAS Aeronautical Journal, Vol. 7, No. 1, p. 3-27, 2016.

[27] Č. Ilić, T. Führer, N. Banavara, M. Abu-Zurayk, G. Einarsson, M. Kruse, J. Himisch, D. Seider, R. Becker, “Comparison of Breguet and ODE evaluation of the cruise mission segment in the context of high-fidelity aircraft MDO,” Notes on Numerical Fluid Mechanics and Multidisciplinary Design, Vol. 132: New Results in Numerical and Experimental Fluid Mechanics X, Contributions to the 19th STAB/DGLR Symposium Munich, Germany, 2014, Springer, 2016.

[28] S. Görtz, Č. Ilić, M. Abu-Zurayk, R. Liepelt, J. Jepsen, T. Führer, R. Becker, J. Scherer, T. Kier, M. Siggel, “Collaborative Multi-Level MDO Process Development and Application To Long-Range Transport Aircraft,” ICAS Paper 2016_0345, in Proceedings of the 30th Congress of the International Council of the Aeronautical Sciences (ICAS), Daejeon, South-Korea, 25-30 September, 2016.

[29] S. Görtz, Č. Ilić, J. Jepsen, M. Leitner, M. Schulze, A. Schuster, J. Scherer, R. Becker, S. Zur, M. Petsch, „Multi-Level MDO of a Long-Range Transport Aircraft Using a Distributed Analysis Framework,” 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA Paper 2017-4326, Denver, 5-9 June, 2017.

[30] Č. Ilić, A. Merle, A. Ronzheimer, M. Abu-Zurayk, J. Jepsen, M. Leitner, M. Schulze, A. Schuster, M. Petsch, S. Gottfried, “Cybermatrix: A novel approach to computationally and collaboration intensive MDO for transport aircraft design,” 21st DGLR-Symposium STAB 2018, Darmstadt, Germany, 2018.

[31] T. Backhaus, S. Gottfried, Č. Ilić, A. Merle, and Arthur Stück, “Integration of High-Fidelity Analysis Tools in MDO Frameworks for HPC,” AIAA Paper, 2019 AIAA Aviation and Aeronautics Forum and Exposition, Dallas, 2019.

[32] P. D. Ciampa, B. Nagel, and AGILE Consortium, “AGILE the Next Generation of Collaborative MDO: Achievements and Open Challenges,” AVIATION 2018, AIAA Paper 2018-3249, Atlanta, 2018.

[33] A. B. Lambe, J. R. R. A. Martins, “Extensions to the Design Structure Matrix for the Description of Multidisciplinary Design, Analysis, and Optimization Processes,” Structural and Multidisciplinary Optimization. 2012; 46(2):273-284.

[34] P. D. Ciampa, E. Moerland, Doreen Seider, Erik Baalbergen, Riccardo Lombardi, and Roberto D'Ippolito, “A Collaborative Architecture supporting AGILE Design of Complex Aeronautics Products,” AVIATION 2017, AIAA Paper 2017-4138, Denver, 2017.

[35] F. Torrigiani, J. Bussemaker, et al, “Design of the Strut Braced Wing Aircraft in the AGILE Collaborative MDO Framework,” ICAS 2018, Belo Horizonte, Brazil.

[36] T. Wunderlich, M. Abu-Zurayk , Č. Ilić, J. Jepsen, M. Schulze, M. Leitner, A. Schuster, S. Dähne, M. Petsch, R.-G. Becker, S. Zur and S. Gottfried, “Overview of collaborative high performance computing-based MDO of transport aircraft in the DLR project VicToria”, Deutscher Luft- und Raumfahrtkongress, Friedrichshafen, Germany, 2018.

[37] Č. Ilić, A. Merle, A. Ronzheimer, M. Abu-Zurayk, J. Jepsen, M. Leitner, M. Schulze, A. Schuster, M. Petsch and S. Gottfried, “Cybermatrix: A novel approach to computationally and collaboration intensive MDO for transport aircraft design”, STAB Symposium Darmstadt, Germany, 2018.

[38] T. Wunderlich, S. Dähne, L. Heinrich and L. Reimer, “Multidisciplinary optimization of an NLF forward swept wing in combination with aeroelastic tailoring using CFRP“, CEAS Aeronautical Journal, Vol. 8, No. 4, p. 673-690, 2017.

[39] T. Wunderlich and S. Dähne, “Aeroelastic tailoring of an NLF forward swept wing“, CEAS Aeronautical Journal, Vol. 8, No. 3, p. 461-479, 2017.

[40] T. Wunderlich and L. Reimer, “Integrated Process Chain for Aerostructural Wing Optimization and Application to an NLF Forward Swept Composite Wing“ in AeroStruct: Enable and Learn How to Integrate Flexibility in Design, Notes on Numerical Fluid Mechanics and Multidisciplinary Design (NNFM), Vol. 138, p. 3-33, Springer 2018.

[41] M. Meinel and G. O. Einarsson. “The FlowSimulator framework for massively parallel CFD applications”. In: PARA 2010 conference, 6-9 June, Reykjavik, Iceland. 2010.

[42] L. Reimer et al. “Multidisciplinary Analysis Workflow with the FlowSimulator”. In: Proceedings of the ONERA Scientific Day 2012—CFD Workflow: Mesh, Solving, Visualizing, ... Ed. by C. Benoit et al. Vol. 19, Ecole Polytechnique, Palaiseau, 2012, pp. 23–30.

[43] S. Freund, F. Heinecke, T. Führer and C. Willberg, “Parametric Model Generation and Sizing of Lightweight Structures for a Multidisciplinary Design Process “ in NAFEMS conference: "Berechnung und Simulation - Anwendungen, Entwicklungen, Trends, 2014.

[44] S. Dähne and L. Heinrich, “Automated Structural Design of Composite Forward Swept Wings“ in AeroStruct: Enable and Learn How to Integrate Flexibility in Design, Notes on Numerical Fluid Mechanics and Multidisciplinary Design (NNFM), Vol. 138, p. 35-38, Springer 2018.

[45] G. A. Wilke, “Variable-Fidelity Methodology for the Aerodynamic Optimization of Helicopter Rotors”, AIAA Journal, published online 24 January 2019, DOI: 10.2514/1.J056486.