Knowledge architecture supporting collaborative MDO in the AGILE paradigm

van Gent, Imco; Ciampa, Pier Davide; Aigner, Benedikt; La Rocca, Gianfranco; Schut, Joost

DOI
10.2514/6.2017-4139

Publication date
2017

Document Version
Accepted author manuscript

Published in
18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference

Knowledge architecture supporting collaborative MDO in the AGILE paradigm

Imco van Gent[*], Pier Davide Ciampa[†], Benedikt Aigner[‡], Jonas Jepsen[§]
Gianfranco La Rocca[¶] and Joost Schut[‖] *AGILE Consortium*

The AGILE project is developing the next generation of development processes, and deploying a collaborative MDO design system, called the AGILE development framework (ADF). Naturally, such a system contains a lot of implicit assumptions on how things should be done and how to exploit different existing technologies. This collection of assumptions and technologies is labeled the 'AGILE Paradigm'. The two main building blocks of this paradigm are the *Collaborative Architecture* and the *Knowledge Architecture*. In essence, these building blocks aim to support large, heterogeneous teams of experts in performing collaborative development in a streamlined and time-effective way. This paper has a focus on the definition of the Knowledge Architecture (KA) as a general conceptual framework which is independent of the aircraft-specific application in AGILE. The KA can be applied to perform collaborative automated design in large, heterogeneous teams for any complex system (e.g. aircraft, automobiles, wind farms). The KA is structured with a multi-level backbone: Development Process layer, Automated Design layer, Design Competence layer. A fourth transverse layer impacting all other layers is the Data & Schemas layer. Each layer has its own set of assumptions and technologies, but more importantly, interfaces between the levels have to be created in order to have a fully interconnected development process from each design competence up to the top-level business process. The hierarchical levels and interfaces are described in this paper as a generalized paradigm. In addition, four support platforms of the KA in the AGILE project are described in more detail: the development process environment, graph-based support in the design problem formulation, visualization of large, complex automated design processes, and design concepts formalizations. Finally, a use case from the AGILE project is mapped on this paradigm to demonstrate the use of the KA and its support platforms in a realistic design case.

Nomenclature

AD	=	Automated Design
ADF	=	AGILE Development Framework
AGILE	=	Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts
CMDOWS	=	Common MDO Workflow Schema
CPACS	=	Common Parametric Aircraft Configuration Schema
DP	=	Development Process
KA	=	Knowledge Architecture
MDO	=	Multidisciplinary Design Optimization
PIDO	=	Process Integration and Design Optimization

[*]Ph.D. Student, Flight Performance and Propulsion Section, Faculty of Aerospace Engineering, TU Delft.
[†]Team lead MDO group, Integrated Aircraft Design Dpt., German Aerospace Center DLR, Hamburg.
[‡]Ph.D. Student, Institute of Aerospace Systems, RWTH Aachen University.
[§]Researcher Engineer, Integrated Aircraft Design Dpt., German Aerospace Center DLR, Hamburg.
[¶]Assistant Professor, Flight Performance and Propulsion Section, Faculty of Aerospace Engineering, TU Delft.
[‖]Chief commercial officer, KE-works.

I. Introduction

Multidisciplinary Design Optimization (MDO) has been a promising design strategy for decades. However, the MDO strategy is not as widely applied as one might have expected at its birth more than three decades ago.[1–3] Instead of speaking only of MDO, one can also see the MDO strategy as a specific case of performing complex automated design (AD). The complexity of the AD can originate from different sources, such as multidisciplinarity, multifidelity, etcetera. Within any design and optimization process three main phases are identified:

1. setup
2. operation
3. solution

Each of these phases comes with its own challenges. In the diagram in the top of Figure 1 a top-level overview of how tackling these challenges would impact the design project time is provided.

In the setup phase the challenge is to integrate different design competences (DCs), often provided by multiple organisations, into a coherent and consistent design process, thereby creating a large, distributed AD process. It is worth noting that already this first challenge is one of the major hurdles for the decision to not use an AD strategy, based on two potential risks. On one side, the need for cross-organizational accessibility of the design competences within a Consortium often result in an infeasible IT solution. Hence, it is difficult and time consuming to set up and execute a workflow containing different design competences present in a Consortium, due to intellectual property and security issues. On the other side, the use of disciplinary models and associated domain-specific knowledge, adopted by different parts of the distributed teams, often results in interpretation mistakes during the design process. This then leads to a lack of trust in the often large AD process that has been set up. Hence, if the AD process is executable (first hurdle is taken), then it could become too large and complex to comprehend it. In a survey of recent MDO-oriented projects that were performed either by DLR (German Aerospace Center) or in a European context, it was found that 60-80% of the project time was used for setting up the first automated chain of multidisciplinary analyses.[4] A similar conclusion was drawn by Flager and Haymaker,[5] who compared a legacy design method to MDO and found that the first design iteration took 133% more time with MDO. Such an increase in time for obtaining merely the first result puts a huge burden on the designer and dramatically increases the risks involved in the project. It is therefore a major hurdle for performing AD.

In the second phase, the operational phase, there is a lack of both detailed understanding and operational flexibility with respect to the complex system that has been set up. This lack of understanding, which originates from missing system level overviews, makes it hard to find any possible inconsistencies in the AD process and to support the identification of design trends and decision making. The operational inflexibility hampers the quick reconfiguration of the process that might be required when progressive insights are obtained. Furthermore, operational inflexibility is encountered when one needs to incorporate new design competences, constraints, or requirements, within an already established process.

Finally, challenges in the solution phase are to find an optimal and robust solution at the end of the AD process, or to find the best solution within an allocated time. These two challenges are mainly concerned with the development of new optimization algorithms and MDO architectures.[6]

Many of the challenges mentioned above are tackled by AGILE (Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts),[7] an EU research project under the funding scheme Horizon 2020 and coordinated by the German Aerospace Center (DLR). AGILE is developing the next generation of aircraft MDO processes, which target significant reductions in aircraft development costs and time to market, leading to cost-effective and greener aircraft solutions. The potential solutions to these challenges are tested in design campaigns, and upon success become part of the *AGILE Paradigm*:[8] a novel design methodology accelerating the deployment of collaborative, large scale design and optimization frameworks, in heterogeneous teams of experts. Main elements of the AGILE paradigm are the collaborative architecture and the knowledge architecture. The former formalizes the collaborative development process, and the elements necessary to set up and deploy a collaborative MDO platform within the entire supply chain.[9] The latter is the subject of this paper: knowledge architecture (KA). The overall conceptual elements composing the AGILE paradigm are shown in Figure 1.

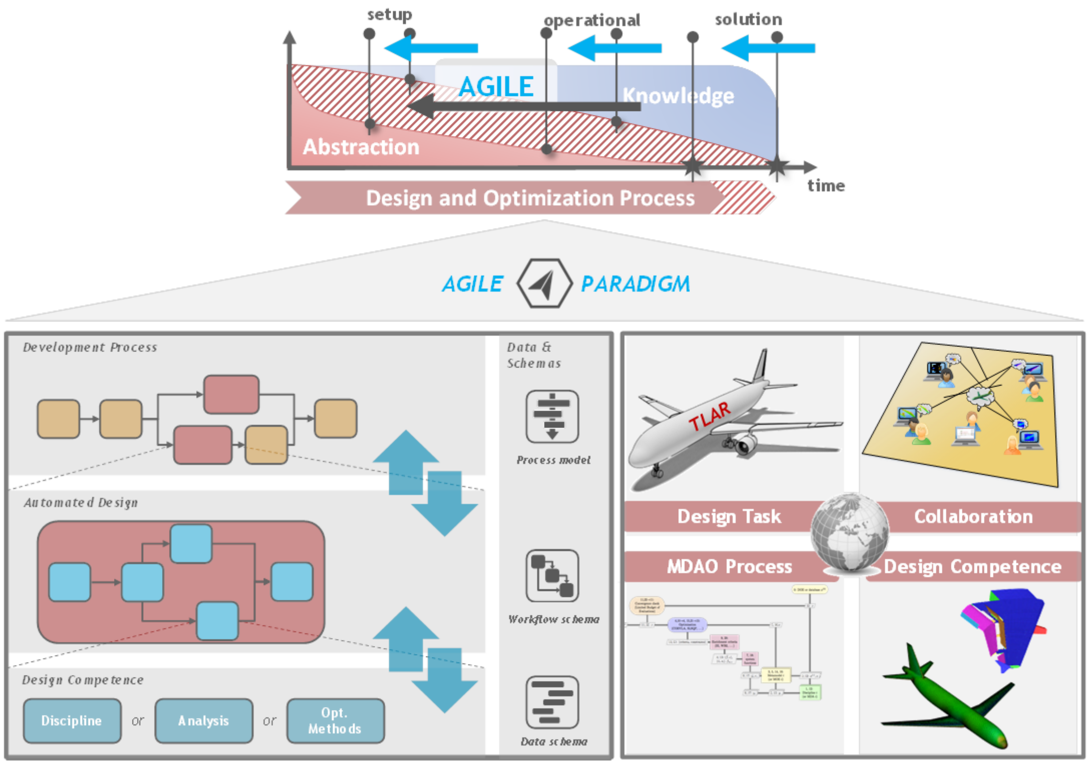


Figure 1. AGILE Paradigm - Conceptual Framework

II. Agile development framework architecture

A top-level overview of the KA is depicted in Figure 2. The different layers of the KA match the different levels of scoping that can also be found in related literature:[10] organisation, framework, tool, and data level. The KA formalizes the multi-level processes that are required to enable the execution of collaborative design and optimization tasks in large, heterogeneous teams of experts. Furthermore, the KA integrates the knowledge, processes, and data at different levels and couples them through interface layers. This formalization of the collaborative MDO process is something that has been on the wish list of the MDO community for a long time, as expressed in the report on the 2011 MDO workshop[11] and the more recent workshop on Complex Systems Integration presented at the ICAS 2016.[12] Among needed enhancements, the shift from structured processes to knowledge modeling is envisioned as a key enabler for the development of the next generation of aerospace products. The panel of experts present in both workshops have estimated a development time for such enhancements between the next 10 to 20 years. The KA developed within the AGILE paradigm, is an effort to provide such a conceptual model.

Although all layers are described in this paper, the focus of current developments is on the middle and bottom layers and the interfaces between them. For example, the interface between the AD and Design Competence (DC) layer attempts to tackle the lack of AD process formalization currently present in collaborative AD. Simply put, this lack of formalization means that IT-wise one would be able to create an AD process, but one looses oversight of what is happening in detail in that same process. The AD process has the risk of becoming a large, complex black box that none of the involved design team members can grasp, inspect, or validate. Using newly developed formalization techniques in AGILE, the goal is to make information on the individual design competences and how they are connected accessible during set-up up and execution of the AD process and in all layers of the KA. This formalization will improve the agility of the design team in three ways:

- reduce the set-up time for large and complex AD processes
- enable systematic inspection and debugging of AD process

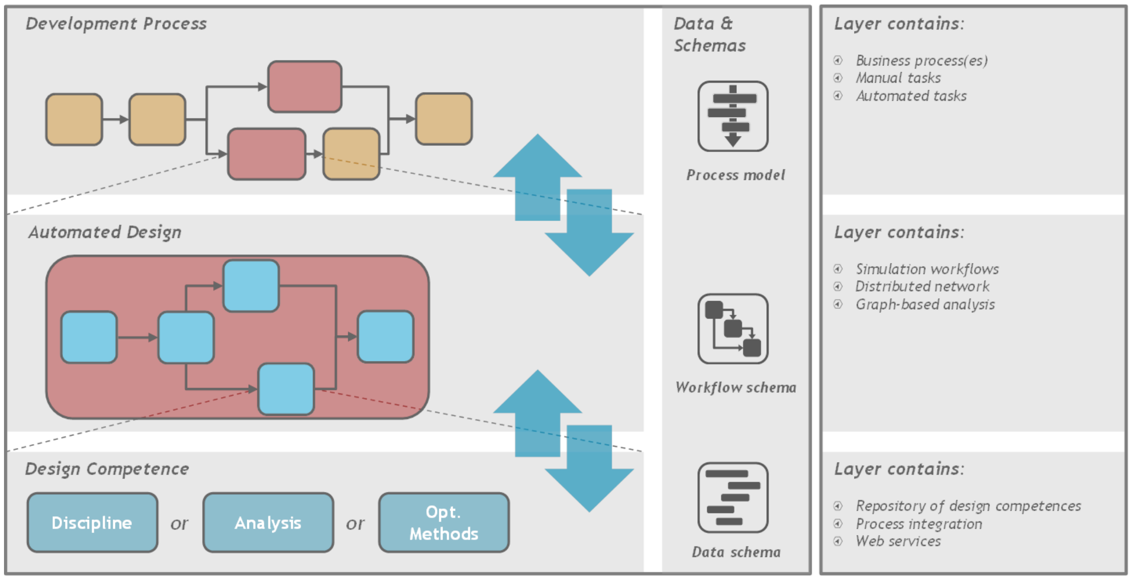


Figure 2. AGILE knowledge architecture to support automated design in large, heterogeneous teams of experts

- allow the manipulation of the AD formalization so that the creation and reconfiguration of optimization strategies can be automated

The three layers of the KA and the interfaces connecting them are discussed in a generalized AGILE paradigm fashion in the sections below. The mapping of an actual use case on this architecture will be done with the case studies described in section IV.

A. Development Process layer

The AGILE development process (DP) is the top layer in the KA. In this layer all activities concerning the design use case at hand are combined. These activities are a combination of manual and automated tasks. Furthermore, knowledge-based technologies[13] are used in this layer to support knowledge acquisition and reuse. The goal of the DP layer is to yield an optimal design at the end of the process.

The business process that is defined in the DP layer is the starting point of the setup and execution of a new multidisciplinary design problem. The two primary functions of the business process level are (1) to support the user in the setup of a design problem and related simulation workflows, and (2) to support the user during the execution of the required automated design process by means of interfaces to manually set up such a process and by retrieving information on the simulation process performance and the design results.

The AD system that is created to perform MDO can be seen as one of the products generated by completing the first iteration of the DP. In the AGILE paradigm, five main steps have been defined for this DP:

I: Define design case and requirements

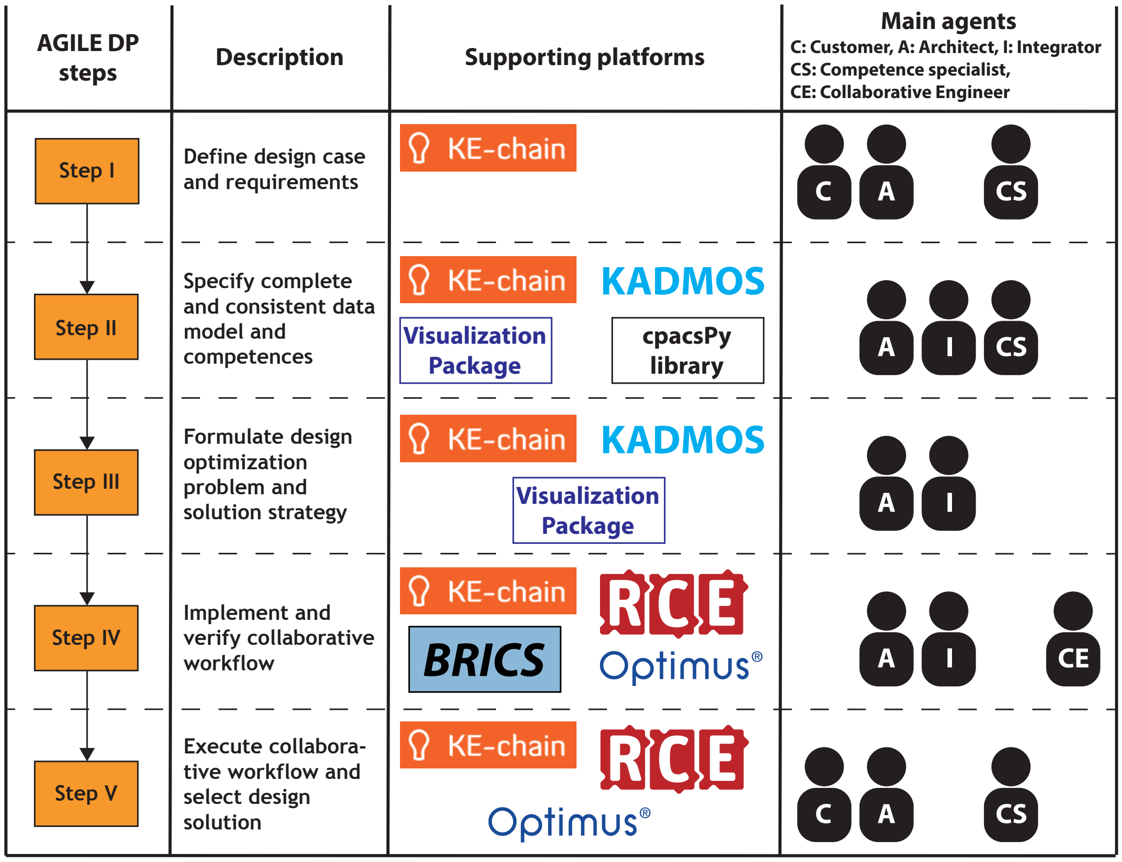
II: Specify complete and consistent data model and competences

III: Formulate design optimization problem and solution strategy

IV: Implement and verify collaborative workflow

V: Execute collaborative workflow and select design solution

The five DP steps above constitute the AGILE Development Framework (ADF) at its highest level and are visualized in Figure 3. Each of the steps uses at least one of the developed ADF supporting platforms and impact the layers below the top DP layer. The ADF supporting platforms will be discussed in Section III and the five-step DP process will be the backbone of the demonstrator in Section IV.



AGILE DP steps	Description	Supporting platforms	Main agents C: Customer, A: Architect, I: Integrator CS: Competence specialist, CE: Collaborative Engineer
Step I	Define design case and requirements	KE-chain	C A CS
Step II	Specify complete and consistent data model and competences	KE-chain, KADMOS, Visualization Package, cpacsPy library	A I CS
Step III	Formulate design optimization problem and solution strategy	KE-chain, KADMOS, Visualization Package	A I
Step IV	Implement and verify collaborative workflow	KE-chain, RCE, BRICS, Optimus®	A I CE
Step V	Execute collabora-tive workflow and select design solution	KE-chain, RCE, Optimus®	C A CS

Figure 3. Five-step AGILE Development Process

It should be noted, how, in the spirit of supporting the collaborative work of heterogeneous sets of experts, different agents are involved in the execution of the the five steps. Five main agents have been identified (and their roles are also more elaborately discussed in Ciampa et al.[9]):

- **Customer:** Customer and primary user of the framework. Responsible for defining the design task, top-level requirements, and available development lead-time. It includes the retrieval of results from the AGILE framework app.
- **Architect:** Responsible for specification of the design case in the AGILE framework, such as collecting the required competences, defining the design phases and the dimensionality of the design space to be explored.
- **Integrator:** Responsible for the deployment of the design and optimization (sub-)processes, and for the management of such processes within the AGILE framework. Intellectual property protection is also administrated.
- **Competence specialist:** Responsible for providing design competence within the framework, such as a simulation for a specific domain, or an optimization service. Specifications of the competences are managed in the AGILE framework app.
- **Collaborative engineer:** Responsible for providing the integration within the framework, necessary to connect the various competences and making them accessible to the framework. It includes the secure integration of software apps in different networks.

The specific involvement of the agents above is indicated in the last column of Figure 3 and will also be further elaborated upon in the demonstrator in Section IV.

B. Automated Design layer

Within the AD layer, the design and optimization process that was formalized in the DP layer is deployed and executed, in order to perform a specific design analysis or optimization. Such a process is assembled by invoking multiple design competences, which are orchestrated according to a specific process architecture.

Within such an orchestration of the design process(es), the design competences are connected via a data channel. Such a process, is configured as an automated simulation workflow, which is deployed by making use of Process Integration and Design Optimization (PIDO) platforms (e.g. Optimus and RCE). Hence, the need to generate (in an automated fashion) a simulation workflow that will guarantee the presence of the required data (within the centralized model or as additional native data format) at the time design competences are executed.

Such a process may regard the overall complex design synthesis task, or the optimization of a detailed component. Within the AGILE paradigm, the communication of cross-organizational competences, and multiple legacy design and optimization processes is accomplished by a 'collaborative platform' that makes use of multiple software environments. Details on the deployment of this collaborative architecture are presented in a companion paper.[9] In the KA the focus in this layer is to have a formalization of the AD process that provides agility to the design team, as such a formalization provides the required interfaces to the upper DP and lower DC layer.

C. Design Competence layer

The design competence layer forms the third and lowest layer in the KA. This layer includes the actual design and analysis tools contributed by the various disciplinary experts. These competences form the core of the AD and they are a prime example of elements in the AD process that induce heterogeneity in the design team. Note that, just like the AD workflow, disciplinary competences are usually also integrated using PIDO platforms, however, by means of micro-workflows, each one including a single design competence. The following assumptions are valid in the AGILE paradigm concerning disciplinary competences:

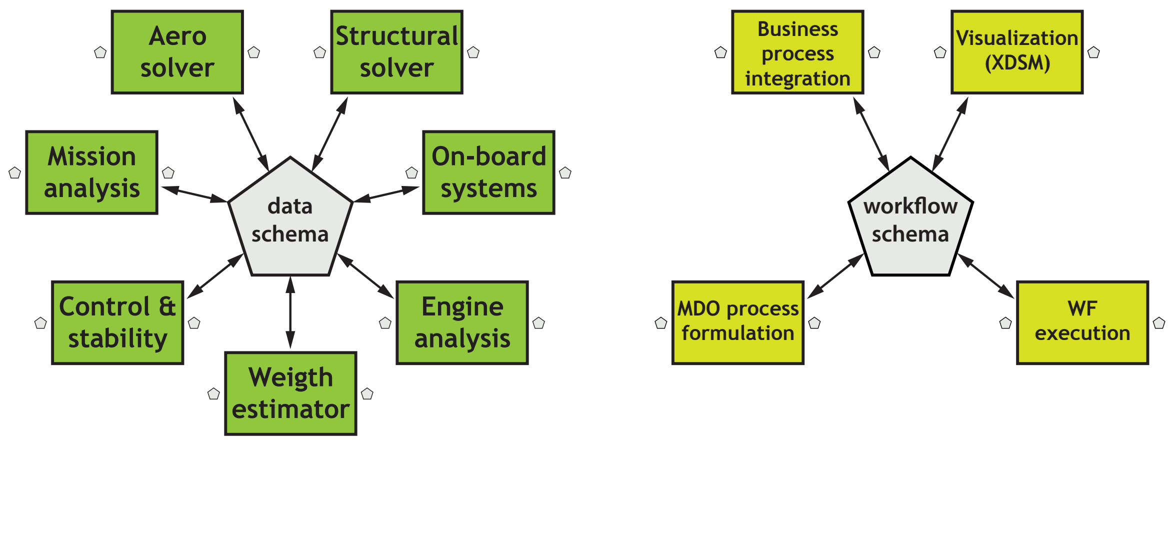
- The heterogeneity of the design team should be respected by:
  - Allowing design competence teams to work out their service in the way they see fit. This means that the competence team can use its own software tools for analysis, as well as their own process integration tools.
  - Allowing competence teams to execute their service on their own system as a web service.
- The integration of the design competence is enabled by:
  - The use of one shared data schema (see details in the next section), which should provide a complete product description and provide entries for competence data (e.g. input and output data of the various design tools).
  - Expecting competence teams to wrap their tools around the central data schema. This means that the service provided by a competence team should use and produce information with respect to the single central data schema.
  - Expecting competence teams to provide standardized service information that might be required by the system integrator (e.g. description, fidelity level, execution time, accuracy)

The above-mentioned assumptions of the design competences are enabled using different technologies that are part of the collaborative architecture.[9] Concerning the KA, the design competences should be stored in a repository using a standardized format.

D. Data & Schemas layer within the KA

Common schemas are used in all layers of the KA, see Figure 2. These schemas are used to enable exchangeability of both design competences and workflows, which ultimately will facilitate a faster integration of the multidisciplinary framework. Two different schemas are required for the exchangeable storage of design competences and workflows. Design competences require a common data schema, such that all the competences can read from and write to a single file. This makes any ad-hoc and direct couplings between DCs unnecessary and results in a minimal amount of interfaces. The data schema is visualized in Figure 4(a). In aircraft design, such a schema is available and called the Common Parametric Aircraft Configuration Schema (CPACS)[14,a].

In addition to the data schema, a workflow schema is also part of the KA. The workflow schema enables the storage of the automated design process in a neutral format. This storage is required in order to be able to benefit from the different ADF support platforms, such as MDO process formulation, visualizations (XDSM), business process integration, and workflow execution. As is shown in Figure 4(b), the workflow schema facilitates the exchangeability of the same workflow between a heterogeneous set of ADF support platforms. The workflow schema, called the Common MDO Workflow Schema (CMDOWS)[15,b], is a more recent development that was initiated in AGILE.



(a) Data schema (b) Workflow schema

Figure 4. Common schemas used in the AGILE KA

The use of common schemas in the KA can also be seen as a constraint that is put on the three KA layers. Individual members of the design team might experience that they are loosing some freedom in defining their own competence, since any data that needs to be exchanged between members of the heterogeneous teams will have to meet schema definitions. However, this schema compliancy is key to enable the definition and execution of collaborative workflows in large, heterogeneous teams of experts.

E. Development Process / Automated Design interface

In Figure 2 two interfaces are indicated implicitly. The lowest interace is the connection between the DP and the AD layers. These layers need to be coupled in two directions. In downward direction (DP $\rightarrow$ AD) the DP tasks can control the AD process by changing settings (e.g. based on a change in design requirements, analysis replacement, or other DP influences). In upward direction (AD $\rightarrow$ DP) the formalization of the AD process needs to be brought to the DP layer. This formalization of the AD can then be used in different ways throughout the DP process, such as the inspection and validation of the AD process in order to grow the design team's confidence in the AD process, and the dissemination and storage of results and lessons learned for future reuse.

F. Automated Design / Design Competence interface

The second interface between the AD and DC layers couples the AD process with the underlying DCs in both directions, see Figure 2. In downward direction (AD $\rightarrow$ DC) the AD process should be able to call the

[a]CPACS is publicly available at: `https://github.com/DLR-LY/CPACS`
[b]CMDOWS is publicly available at: `http://cmdows-repo.agile-project.eu`

different DCs as required for the defined AD process. To respect the individuality of the design competence team, the AD process should only be allowed to call a design competence, while leaving the actual execution and feedback up to the competence specialist. Hence, the AD process should not be allowed to demand the execution of the DC in ways that are outside the 'comfort zone' of the DC team.

The upward direction in this interface (DC $\rightarrow$ AD) entails that the full definition of all design competences is made available to the AD layer. This repository of design competences should contain the central data schema, the information used and produced by each DC with respect to this schema, and meta-data on the design competence that might be relevant for the AD process definition (e.g. how to call, average execution time, fidelity level, accuracy).

Within the AGILE paradigm, this interface has to overcome two challenges. As highlighted in the introduction, the assembly of a consistent set of models and processes, which characterizes in a specific stage of the development both the complex product and the methodologies employed for its design, is an extremely challenging task. Although a central data schema may be used for the exchange of data between the design competences, usually design competences use the description from the central data format and convert it into an internal representation of the data. Hence, the consistency among competences highly depends on a series of interpretation steps from the common representation to the design competence specific representations. The interpretation usually differs with the competences (such as for different disciplines), and with the level of complexity which is retained within each of the abstractions for each of the model's representations. Therefore, when multiple design competences are integrated within a design process, it is necessary to guarantee not only a definition of a common model, but also the definition of the common 'concepts' which are shared by the different model abstractions. Such a concept layer will enhance the effective setup of a consistent design process integrating multiple design competences. This concept layer is the first challenge to overcome. It would enable the use of a common concept in the AD process, while this concept is not explicitly defined in the data schema used by the design competence.

Another challenge to overcome in this interface, is the capability to bridge the formalizations at the DC layer (inputs, outputs, common model), with the formalization at the AD layer (e.g. a simulation workflow). Such a capability would be the first step to support the automatic setup of design and optimization processes according to the design competences that are available within a project. Hence, one would need to couple the DC formalizations in a central system, such that this system can be used to create, debug, and manipulate the repository of design competences, ultimately resulting in executable simulation workflows for the AD layer and human-readable AD process overviews for the DP layer.

III. Support platforms for the ADF

The different layers of the ADF are connected using ADF support platforms. This section presents four different support platforms currently under development in the AGILE project, namely:

- Development process environment
- Graph-based support in the design problem formulation
- Visualization of large, complex AD processes
- Design concepts formalizations

These platforms create connections between the different layers of the KA. The relation with the KA are depicted in the different figures by using the same layers as given in Figure 2 and/or the same steps as indicated in Figure 3.

A. Development process environment

The development process environment of AGILE is composed of two aspects; a data model and process environment. Both are defined specifically with MDO problems in mind but could be used outside these boundaries.

The MDO data model of the DP environment uses the CMDOWS format. The data model function is a common interface within and beyond the AGILE DP. The data model contains all key information used to define, use, and store MDO problems and workflows. CMDOWS is an XML-based workflow schema and is independent of the data model of the MDO problem. For more information on the CMDOWS data model the reader is referred to Van Gent et al.[15] and the publicly available repository.

The AGILE development process depicted in Figure 3 is implemented with the KE-chain platform.[16] The DP environment's emphasis is on the DP layer of the ADF. Its function is to provide MDO teams with means to manage the life-cycle of the MDO study being performed. KE-chain enables the MDO team to define, adapt, and reuse MDO studies. Key aspects of the environment are multi-user web-based collaboration, shared master data model, shared master process model, data configuration management and storage and reuse. The KE-chain platform communicates with the other layers by means of the CMDOWS data model, which is adjusted and enriched during the entire development process

B. Graph-based support in the design problem formulation

One of the challenges to be tackled in AGILE is the fact that MDO problems are growing in size and complexity.[17] The risk involved in this is that the created AD process becomes a black box that is not, and cannot, be understood by the design team due to its sheer size and complexity. In AGILE, a graph-based system is under development for this purpose. In essence, graphs are nothing but a special way to store data using nodes and edges, however, the storage format and available analyses match very well with the challenges that need to tackled in AGILE. The system is called KADMOS: Knowledge- and graph-based Agile Design for Multidisciplinary Optimization System. KADMOS aims at improving the AGILE DP process by supporting three tasks:

- **Create:** The formalization of large, complex AD processes.
- **Inspect & debug:** Enabling different experts (i.e. disciplinary specialists, integrators) to inspect sections of the model that are relevant to them, in order to validate the model, thereby increasing the level of trust in the model.
- **Manipulate:** Automatically manipulating its own content using graph-based analysis. These computerized manipulations reduce the chances of errors and inconsistencies in the model since repetitive error-prone human tasks are replaced. Simple manipulations would be the processing of changes in input or output variables of a design competence. More advanced manipulations are the automated transformations required to go from the repository of competences to an executable AD workflow according to a given MDO architecture.[6]

A system overview of KADMOS is depicted in Figure 5. The graphs created and manipulated by KADMOS use or affect all layers of the KA. KADMOS is used in the third step of the ADF development process. Where there would normally be a gap between the formalization (i.e. blueprint) of the AD process to be executed and the actual executable collaborative workflow, KADMOS enables a design team to go, in an agile way, from a repository of DCs to a collaborative workflow in Optimus or RCE. Hence, in Figure 2 KADMOS provides the upward DC/AD interface for workflow formalization.

As shown in the second column of Figure 5, the recently developed CMDOWS format is used to exchange workflow data between the different ADF process steps. One of the results of step II is the definition of the repository of DCs, including their input and output links with the data schema, which is stored in a CMDOWS file. As a result of step III, an enriched CMDOWS file is created that contains the full neutral definition of the AD process to be executed. In each substep in step III, different graphs are created using graph manipulation algorithms. Currently, KADMOS supports the automatic workflow creation for two different MDO architectures[6] (Multidisciplinary Feasible (MDF) and Individual Discipline Feasible (IDF)) and can also impose design analysis architectures such as design point convergence and design of experiments. The full details of KADMOS are described in a separate paper.[18] The visualizations mentioned in the last column of Figure 5 are discussed in the next section.

C. Visualization of large, complex AD processes

The design and optimization cases in the large and complex MDO systems, which are created in AGILE are based on multiple disciplinary design and analysis tools provided by the different project partners. It is therefore almost impossible for an MDO architect setting up a problem solution workflow to grasp all underlying, relevant information on these systems manually and all at once. Therefore, it is important to

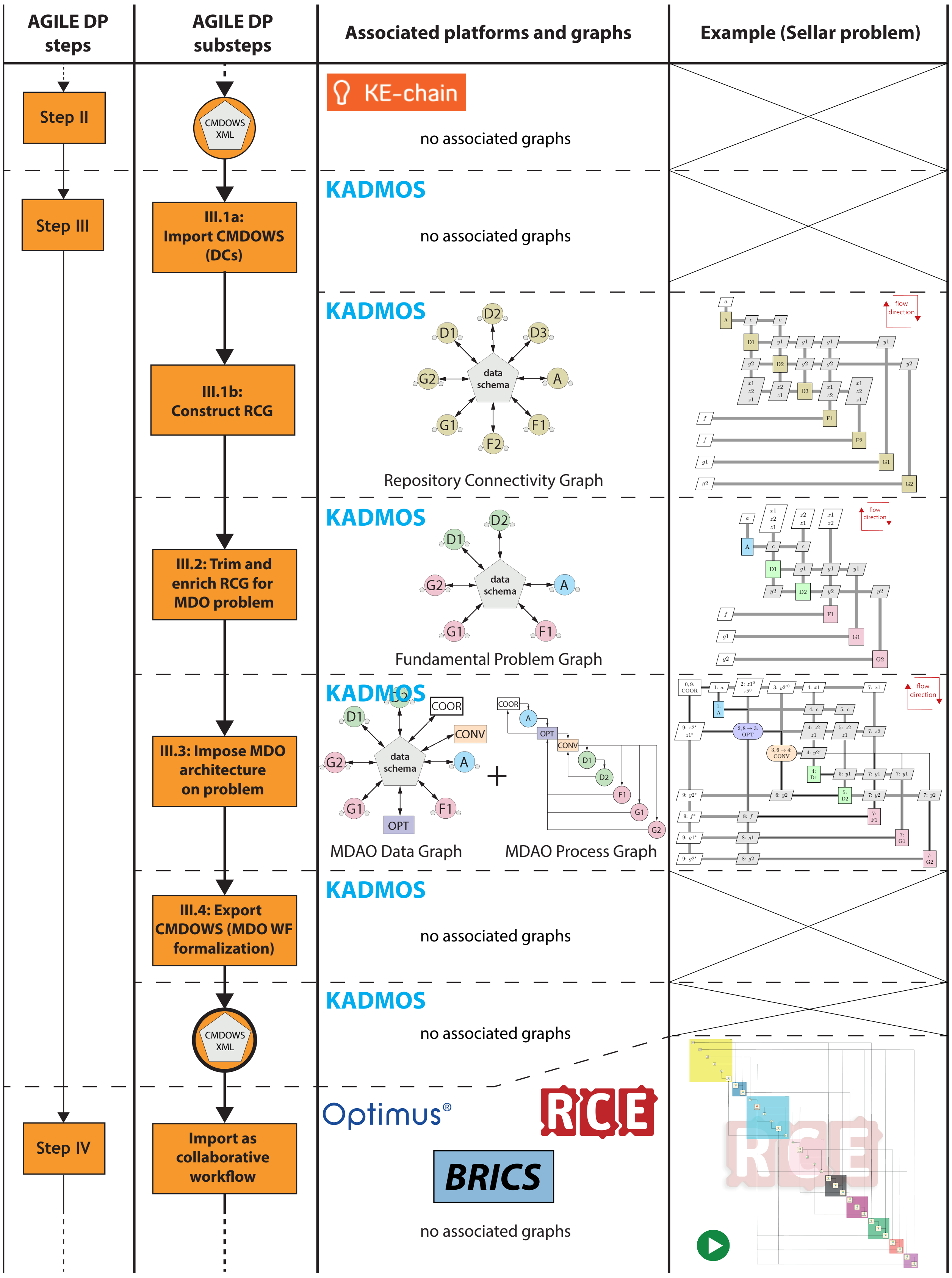


Figure 5. Top-level overview of KADMOS and its relation to the ADF development process (all visualizations are based on the Sellar problem[19])

provide this relevant information - which is captured and processed automatically by the software system KADMOS - in a human-intelligible way.

Hence, a significant aspect in the DP/AD interface is data and process visualization, which is for instance used in order to support the MDO architect in the task of MDO problem formulation. Some of the commonly used diagrams in the field are N2 charts,[20] functional dependency tables (FDT)[21] and (extended) design structure matrices (XDSM).[22,23] With the help of these techniques it is possible to capture the full description of an MDO system, including system couplings and service execution order. The drawback of the above-mentioned, conventional visualizations is twofold. First of all, the generation of formal MDO system visualizations by means of the aforementioned methods is generally a manual and highly complex task. On top of that those methods are not readily applicable for large and complex problems: their static form is simply not suitable to render large system representations on current display means, such as reports and computer display.

Therefore, in AGILE, an advanced dynamic visualization package is used, in which the knowledge on the MDO systems is translated into human-readable plots and diagrams. The package makes use of already existing visualization techniques such as XDSMs, Sankey diagrams and hierarchical edge bundles.[24] These visualizations are enhanced with help of the open-source JavaScript library D3.js, which is specifically tailored for dynamically visualizing data.[25] D3.js therefore enables the desired capabilities for the AGILE KA. While D3.js comes with a large amount of predefined visualizations, it is also possible to visualize data in any desired fashion. The main advantage of using D3.js is the dynamic accessibility of the visualizations. Large and complex XDSMs can for instance be expanded or collapsed in order to set the focus on certain aspects of the MDO system. Couplings between services or variable inter-dependencies can be analyzed in comprehensible diagrams, in which it is possible to dynamically reveal detailed information (e.g. via mouse click/hovering), that would either not be visualized at all, or too confusing to understand when visualized all at once. Therefore, the visualization package developed within the AGILE framework can be used as a debugging environment, in which an MDO architect is able to examine the created architectures, review the steps of the product development process and detect possible issues. The visualizations are part of the DP/AD interface. Possible XDSM visualizations for a small MDO model are shown in the last column of Figure 5, others are presented later in Section IV of this paper and in Van Gent et al.[26]

D. Design concepts formalization

In AGILE, CPACS is adopted as common language, and it is a fundamental element to enable the distributed multidisciplinary design process integrating competences from multiple partners. Despite of the central data format, every discipline has its own way of representing the aircraft. This is essential for efficient data handling and problem description, as the efficiency with which a problem can be described and solved highly depends on the chosen knowledge representation. Consequently CPACS data always has to be pre- and postprocessed by every disciplinary competence.

Figure 6 shows an exemplary conversion of a high fidelity wing, as stored in the central data format, into a concept of a single trapezoid wing. During the conversion a reduced number of parameter values for the simple wing concept need to be determined which represent the detailed wing as close as possible (Figure 7). At this point decisions need to be made on how the parameters from the low fidelity wing concept are defined based on the high fidelity shape. These decisions are essential for flexibility with which the competence can be used at a later stage. A use case for which any of the assumptions is violated cannot make use of the competence or the competence needs to be adjusted according to the new use case. Therefore this step should not be taken lightly. In order to check if all assumptions are valid, every competence needs to provide a complete list of all assumptions made internally to be passed to the integrator. Only with the complete list of assumptions the integrator can ensure that all the concepts which are used by the competences are actually valid for the every individual use case.

Unfortunately the interpretation of the data and the assumptions made during the conversion to an internal model are rarely available in an explicit form. Violation of assumptions leads to faulty results and when not recognized to an invalid design. The potential for such issues is particularly high in multi-fidelity processes where conversions between high fidelity and low fidelity representations are frequently done.

For dealing with these less obvious compatibility issues a layered approach for the setup of a complex multidisciplinary and multi-fidelity design process is formulated which supports the user in avoiding hidden inconsistencies. Such a layer representation has been defined by Jepsen et al.[27] and is illustrated in Figure 8.

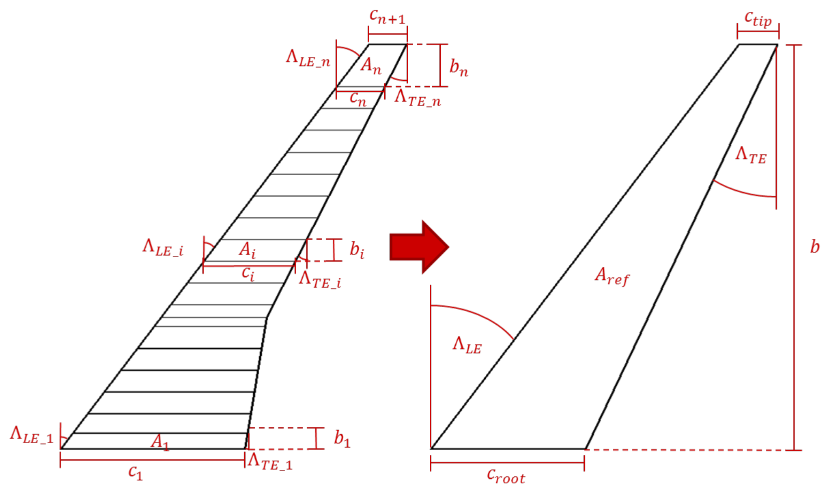


Figure 6. Converting wing models

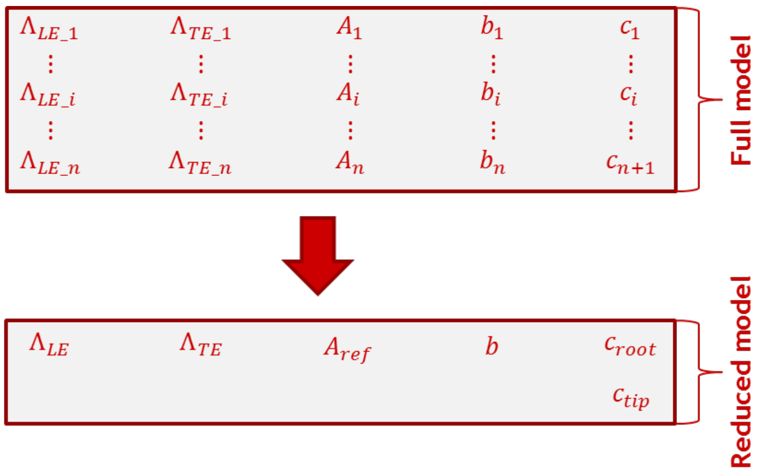

Λ_{LE_1} Λ_{TE_1} A_1 b_1 c_1
Λ_{LE_i} Λ_{TE_i} A_i b_i c_i
Λ_{LE_n} Λ_{TE_n} A_n b_n c_{n+1}
Full model
Λ_{LE} Λ_{TE} A_{ref} b c_{root}
c_{tip}
Reduced model

Figure 7. Reducing number of parameters for a simple wing model

- **Data & Schemas layer:** contains the description of the model as standard representation (in AGILE a valid CPACS definition).
- **Concept interface layer:** provides the abstraction from the CPACS model properties to the model representation properties ad-hoc for the design competence application.
- **DC Layer:** contains the description of the model as required by the AD process.

The additional concept interface layer serves as a means to communicate all assumptions used within the individual competences. By referring to predefined concepts with clearly documented assumptions, concept providers are able to give concise information on the internal assumptions and thus allow the integrator to check compatibility between all the competences involved in a specific design process. When providing an implementation of formalized concepts together with an extensive documentation, the effort to wrap a design competence can be heavily reduced. The implementation of means for extracting concept data from the central data format and writing data back to the same allows competence providers to use existing concepts for providing an interface to the centralized data format.

For the CPACS-based environment in AGILE, the cpacsPy library was used to develop the morphing competence which modifies wing geometry from a set of design parameters which are not explicitly de-

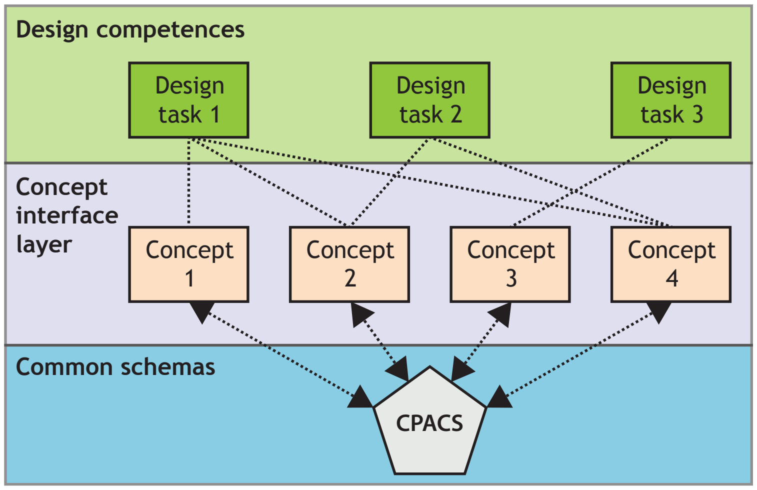


Figure 8. Design concepts formalization

fined/directly accessible in CPACS. A detailed description on how the cpacsPy was used to develop the morphing competence is given in section IV.B.2.

In AGILE, the layers can be passed through either way, top-down for converting from the application data to CPACS, or bottom-up for converting CPACS data into the application's internal representation.

IV. Demonstrator

The first case study used to demonstrate the KA is based on the result of the first design campaign performed within the AGILE project. This design campaign targets the conceptual and preliminary design of a medium-range twin-engine jet airliner. Figure 9 provides an initial geometry of the aircraft that has not yet been optimized. A collection of CPACS-compliant design competences has been used to set up an optimization task of the wing using aerodynamic, structural and mission performance analysis. In the following sections each of the five AGILE DP process steps (Figure 3) will be described to demonstrate how the KA comes into play in a realistic design case. For each step it is here detailed: the main function description and the associated deliverables, and the supporting platforms and main agents[9] involved.

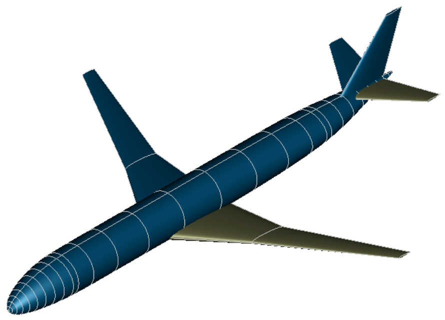


Figure 9. The AGILE conventional aircraft from the first design campaign

A. Step I: Define design case and requirements

Function: In the first step, the design and optimization use case is identified and the requirements for the development process are defined. The step includes the selection of the design capabilities (among those available in the repository), and the definition of the overall boundary conditions affecting the resolution of the design problem, such as the available lead time. The step is also in charge for the management of requirements, and design choices which may evolve and change during the development process.

Agents: Customer, Architect, Competence Specialists

Supporting platforms: The step is completely performed in the AGILE DP app implemented in the supporting platform KE-chain.

Deliverable: A formalized description of the use case persistently stored and browsable in KE-chain.

Substeps: The step is divided into three substeps, in which the following actions are performed:

Step I.1 Definition of the design systems' requirements. The main information on the use case is collected from the involved customers, and translated by the architect into a set of requirements. These include performance specifications for the product and process, and corresponding means of compliance are defined.

Step I.2 Definition of competences. The set of competences available for the resolution of the use case are collected by the architect and the competence specialists. The information provided, support the selection of the necessary competences, and the identification of the missing ones. At this stage the set of parameters which are relevant in the design process is assembled, and their roles within the design and optimization task (e.g. objective of the design) is assigned.

Step I.3 Requirements management. The architect keeps track of the requirements compliance during the execution of the entire development process.

B. Step II: Specify complete and consistent data model and competences

Function: The purpose of the second step is to define a complete, consistent and compliant design process. Therefore the parameters and competences assembled in the first step are formalised and adapted such that at least one complete process can be obtained. Therefore the common data model of the parameters and competences has to be defined (here CPACS), and missing links in the workflow have to be complemented by changing the parameters and competences.

Agents: Architect, Integrator, Competence Specialists

Supporting platforms: Development process environment KE-chain, the visualization package (through KADMOS), and the design concepts formalization platform.

Deliverable: A CMDOWS file containing a complete and consistent repository of DCs which are compliant with CPACS.

Substeps: Four substeps have been identified, namely:

Step II.1 First, the integrator should import CPACS-compliant competences into the repository. These CPACS-compliant DCs should be made available by the competence specialists. The data models are integrated in a KE-chain project.[16] The modelling environment of a KE-chain project is used to get the first version of the master data and process models. The result is an initial master data model and master process model.

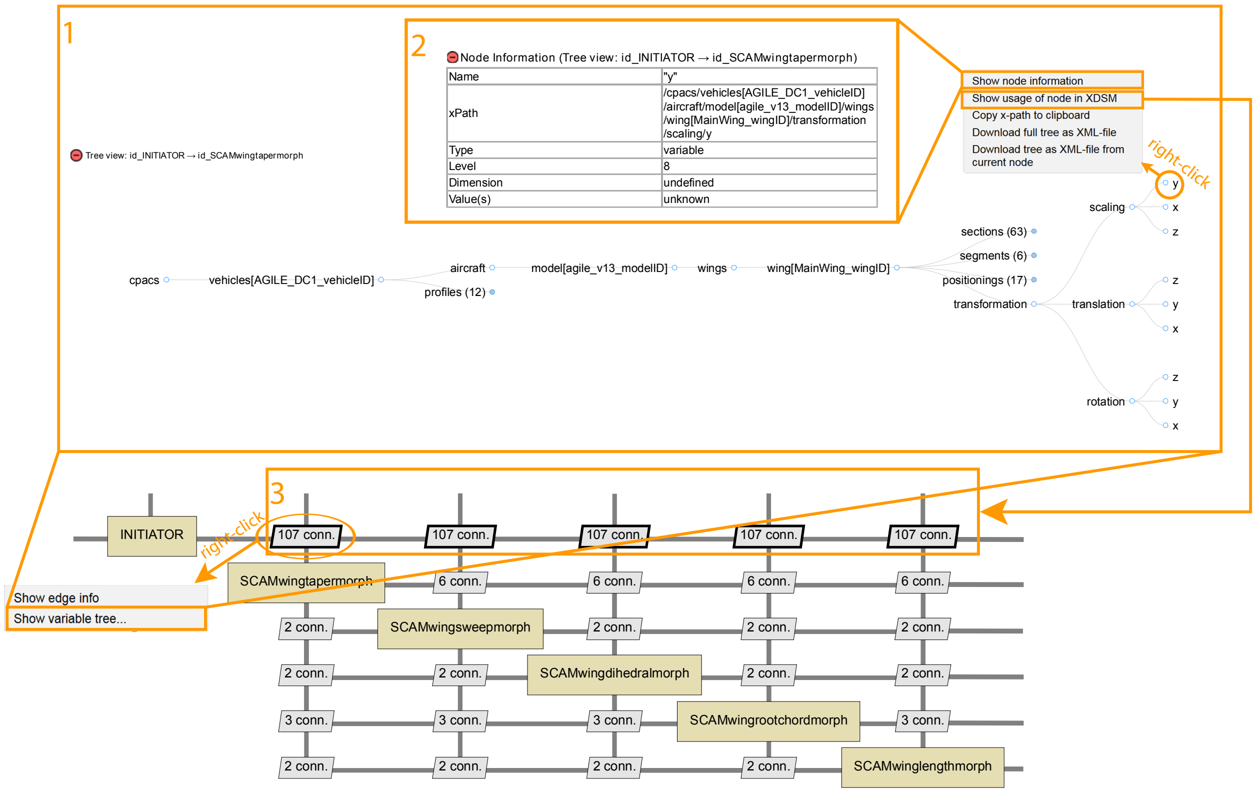
Step II.2 The architect can now edit the DC repository in consultation with the competence specialists. Thereby they both inspect and adapt the data and process models. The modelling environment of the KE-chain project is used to inspect and adapt the data and process models. The KE-chain environment provides the user with a debug environment to find missing links in the process definition obtained from step II.1. This definition is complemented until a complete, consistent and compliant master data model and master process model are obtained.

Step II.3 The integrator can now generate the CMDOWS file. The complete, consistent and compliant master data model and master process model are stored in the CMDOWS format. This CMDOWS file of the repository of DCs can be visualized using the visualization package, as is described after these steps.

Step II.4 Finally, the competence specialist should specify competence execution and accessibility. This final step is meant to prepare all information required for execution and inspection of the MDO problem in steps IV and V. This is currently managed by keeping track of the status of each individual competence. A more detailed implementation of this substep is foreseen.

1. *Use of visualization package in step II*

Once the CMDOWS file for the data model and competences is available, a visualization package can automatically be created. This package is created through KADMOS. At this step of the case study, the underlying data processed by KADMOS contains the maximum connectivity between the different competences - i.e. the interfaces between the competences through the data model - without taking into account the specifics of the design problem yet. The corresponding so-called Repository Connectivity Graph (RCG) can be examined in the visualization package via a web browser. An extraction of an XDSM for the RCG of the wing design use case is shown in Figure 10, including some of the visualization capabilities.



Node Information (Tree view: id_INITIATOR → id_SCAMwingtapermorph)

Name	"y"
xPath	/cpacs/vehicles[AGILE_DC1_vehicleID]/aircraft/model[agile_v13_modelID]/wings/wing[MainWing_wingID]/transformation/scaling/y
Type	variable
Level	8
Dimension	undefined
Value(s)	unknown

Figure 10. Extraction of the visualization package XDSM for the the wing design use case RCG

Note that in the actual visualization package, the full XDSM for this RCG is significantly larger than the extraction shown in the figure with more than 20 competences and over 100 sets of data connections in total. Also the orange overlay frames in Figure 10 are not visible in the actual package and have been included by the authors to emphasize the visualization capabilities. Within the scope of the AGILE project the presented visualization capabilities have already proven to effectively assist the MDO architect in the problem formulation process.[28]

2. *Implementation of the design concepts formalization in step II*

A special DC required in this step is the morphing competence which is based on the design concepts formalization platform. The development of the morphing competence was performed by using the approach of the concept interface layer as described in Section III.D. By making use of existing parametrized wing and wing section concepts the morphing competence modifies the wing shape according to chosen design parameters. One of the design parameters is the **aspect ratio** of the wing, which for a multi-segment wing is defined as "*the square of the sum of the individual segment spans, divided by the cumulated reference area*".

Since the aspect ratio is not explicitly defined in CPACS, it needs to be extracted from the wings geometric description. Jepsen et al.[29] have shown that for a specific box-wing configuration a change in aspect ratio effected 153 CPACS parameters. All the coordinate transformations and data handling required for the changes would have to be implemented in the morphing tool. By using the wing concept provided by the cpacsPy library all that is required for modifying the aspect ratio is a single function call. The process of handling the CPACS data is fully transparent to the user, who can focus on the actual design problem, in parameters suitable to the problem domain, rather than on converting data from and to CPACS. Thus the cpacsPy library does not require users to have a deeper knowledge on the CPACS data model. Competences can be wrapped to CPACS by simply using suitable concept parameters. Additionally assumptions made during the conversion are clearly documented through the concept definition and can be provided to the integrator to ensure consistency throughout the design process. Admittedly the conversion between CPACS and the disciplinary concepts needs to be defined and implemented at some point. Once the library includes concepts for the most common disciplines it will lead to a reduced effort in developing new competences, remove potential interpretation errors and helps avoiding inconsistencies by explicit mentioning of assumptions made during the conversion. In case of the morphing competence concept parameters for all the design parameters could be found in the cpacsPy library, reducing its development time and effort significantly.

C. Step III: Formulate design optimization problem and solution strategy

Function: The goal of this step is to go from a repository of design competences to an MDO process formulation (blueprint) of the collaborative workflow to be executed in step IV.

Agents: Architect, Integrator

Supporting platforms: As also indicated in Figure 5, step III is fully controlled by KADMOS. However the different substeps have been integrated in the KE-chain DP environment, making KE-chain serve as a graphical user interface (GUI) for the KADMOS Python package. The visualization package is also used in this step.

Deliverable: A CMDOWS file containing the neutral formulation of the required MDO process (including visualizations).

Substeps (see also Figure 5): All substeps are performed by the architect unless indicated otherwise:

Step III.1 First the repository connectivity graph (RCG) is constructed in KADMOS. This is simply a translation of the XML-based CMDOWS format that is provided by step II to a graph-based KADMOS graph format. The RCG is a web of data that indicates system inputs, outputs and couplings for a repository of DCs.

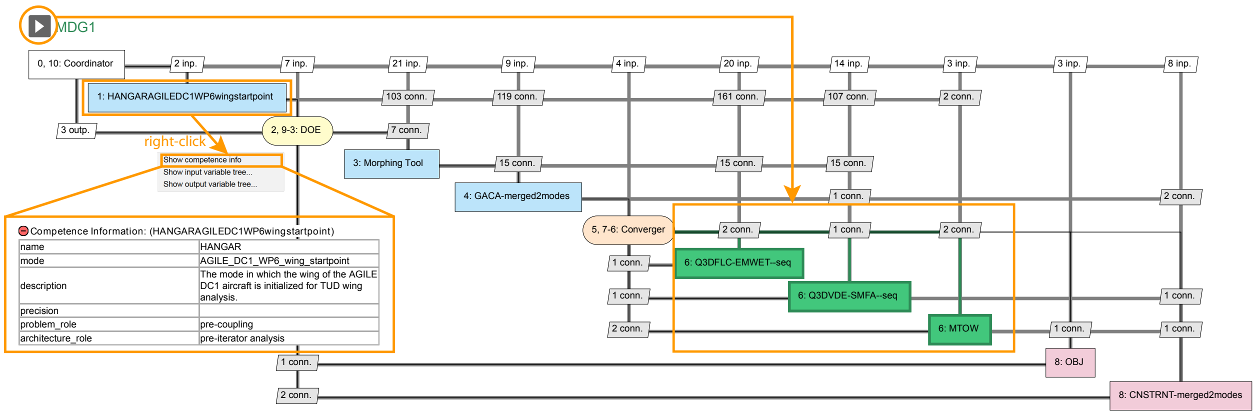
Step III.2 The RCG needs to be trimmed and enriched in order to arrive at the fundamental problem graph (FPG). The FPG is a subset of the RCG. Trimming of the RCG can be done in many different ways (e.g. removal and contraction of graph nodes), the goal of it is to get the smallest possible graph that contains all the DCs and their data connections necessary to solve the MDO problem at hand. The enrichment is performed by adding attributes to certain graph nodes to indicate that they have a special role in the MDO problem, such as design variables, objective values, and constraint values.

Step III.3 In this substep the MDO architecture of choice can be imposed on the FPG. In this case a DOE architecture is chosen, see Figure 11. Based on the FPG, KADMOS automatically sorts the different DCs and implements the additional blocks for the architecture (Coordinator, DOE and Converger in Figure 11). The KADMOS graph algorithms will create two graphs: a data and a process graph. Together these graphs contain the neutral definition of the collaborative workflow to be executed. The combination of the data and process graph can be visualized using the XDSM.

Step III.4 Finally the neutral workflow definition is stored in a CMDOWS file and checked by the integrator.

1. *Use of visualization package in step III*

The CMDOWS files containing the FPG as well as the MDAO data and process graphs (MDPG) is processed to the visualization package. Each MDPG for instance contains an MDO architecture and a solution strategy that combines the information of the data connections and the characteristics of the MDO process (competence execution order, iterations, etc.). Figure 11 shows an MDPG of a Design Of Experiments (DOE) with an internal Jacobi converger for the wing design use case and some of the underlying dynamic visualization capabilities.



Competence Information: (HANGARAGILEDC1WP6wingstartpoint)

name	HANGAR
mode	AGILE_DC1_WP6_wing_startpoint
description	The mode in which the wing of the AGILE DC1 aircraft is initialized for TUD wing analysis.
precision	
problem_role	pre-coupling
architecture_role	pre-iterator analysis

Figure 11. XDSM of a DOE with Jacobi converger for the wing design use case

In comparison to the XDSM for the RCG presented in step II the amount of competences and data connections have been narrowed down for the specific use case. Additionally, architecture-specific competences such as the DOE (Design Of Experiments) and the Converger, which could also be one of the competences provided by one of the partners in the Consortium, have been automatically included by KADMOS. Note that the visualization package is solely developed for examination of MDAO architectures. No actual competences can be executed from here, which is part of step IV of the product development process. As in Figure 10 the orange overlay frames in the above figure have been added to emphasize the capabilities of the visualization package.

Within the process of reconfiguring and adding new MDAO Problem Solutions to a design case with KADMOS, the number of different MDPGs for the same design problem can become quite large. One of the advantages of the visualization package is the fact that all of the graphs produced by KADMOS are eventually merged into one single package and can be selected via a drop-down list in the main menu. That way, an MDO architect can easily compare different architecture types for a specific design problem.

D. Step IV: Implement and verify AGILE framework app

Function: The CMDOWS file from step III should be used to automatically translate the potentially large and complex workflow in the CMDOWS file into an executable workflow in PIDO platform.

Agents: Architect, Integrator, Collaborative Engineer

Supporting platforms: In AGILE the PIDO platforms RCE and Optimus are used for this purpose. More information on the deployment of collaborative workflows can be found in another AGILE paper.[9] The automatic import of a CMDOWS file into RCE and Optimus is a recent development that is still in progress.

Deliverable: A collaborative workflow that can be executed using a PIDO platform.

E. Step V: Execute collaborative workflow and select design solution

Function: After a successful CMDOWS import, the collaborative workflow can be executed in order to receive the first analysis results. The design solutions generated in the previous step, captured in the CPACS format, are provided to the DP environment. The pupose of the step is to prepare the inspection environment and generate the required design results. This step is still under development.

Agents: Customer, Architect, Competence Specialist

Supporting platforms: PIDO platforms RCE and Optimus are used for execution of the collaborative workflows.

Deliverable: Result of collaborative workflow execution.

Note that in a realistic design case, the found design solution will probably trigger an iteration which goes back to one of the earlier steps in the AFD process. Additional requirements might be found (step I), an additional tool must be integrated (step II), or the collaborative workflow might need to be reconfigured using a different MDO architecture (step III). Since with the ADF all steps are integrated within the top-level DP environment, the process that has been set up in the DP environment acts as a custom-made 'AGILE framework app' that can be used to reconfigure the MDO use case and its associated collaborative workflow.

V. Conclusions

One of the main conceptual elements of the AGILE paradigm has been presented: the knowledge architecture, see Figure 2. The four different layers and their interfaces have been defined and more details have been provided on four ADF support platforms: the development process environment, graph-based support in the design problem formulation, visualization of large, complex AD processes, and design concepts formalizations. In addition, the five-step AGILE development process has been presented as the backbone of performing MDO is large, heterogeneous teams of experts. The KA and its support platforms have been demonstrated using the five-step DP with an MDO use case of a conventional aircraft wing using a repository of CPACS-compliant design competences.

In the current stage of AGILE the first three steps of the ADF have been fully implemented and tested. The development of the last two steps is currently in progress, however, the approach is already speeding up the setup of current design campaigns performed in AGILE by reducing the effort required to identify inconsistencies and the ability to reconfigure the design process on the fly. The ADF as a whole will be further developed and tested in the MDO use cases in the final year of AGILE concerning the MDO of a blended wing body, box-wing, and other unconventional aircraft configurations.

The KA presented in this paper has been developed within the context of aircraft design problems, but it can be generically applied to develop any complex product using MDO in large, heterogeneous teams of experts (e.g. aircraft, automobiles, wind farms). It is part of the novel methodology that will be made available during the AGILE project.

Acknowledgments

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

References

[1]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, pp. 615.

[4]Ciampa, P. D. and Nagel, B., "Towards the 3rd generation MDO collaboration Environment," *30th Congress of the International Council of the Aeronautical Sciences*, 2016.

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

[6]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.

[7]"AGILE portal: `http://www.agile-project.eu`," 2016.

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

[9]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.

[10]Berends, J. and van Tooren, M., "Service oriented concurrent engineering with hybrid teams using a multi-agent task environment," *Collaborative Product and Service Life Cycle Management for a Sustainable World*, Springer, 2008, pp. 387–400.

[11]Simpson, T. W. and Martins, J. R. R. A., "Multidisciplinary design optimization for complex engineered systems: report from a national science foundation workshop," *Journal of Mechanical Design*, Vol. 133, No. 10, 2011, pp. 101002.

[12]Ying, S. X., "Report from ICAS workshop on complex systems integration in aeronautics," *30th Congress of the International Council of the Aeronautical Sciences*, 2016.

[13]Milton, N., *Knowledge technologies*, Vol. 3, Polimetrica S.a.s., 2008.

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

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

[16]"KE-chain website: `https://www.ke-chain.com/platform`," 2017.

[17]Pate, D. J., Gray, J., and German, B. J., "A graph theoretic approach to problem formulation for multidisciplinary design analysis and optimization," *Structural and Multidisciplinary Optimization*, Vol. 49, No. 5, 2014, pp. 743–760.

[18]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.

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

[20]Lano, R. J., "The N2 Chart," *TRW Software Series*, 1977.

[21]Wagner, T. C. and Papalambros, P. Y., "General framework for decomposition analysis in optimal design," *ASME Advances in Design Automation - Volume 2*, Vol. 65-2, 1993, pp. 315–325.

[22]Steward, D. V., "The design structure system: a method for managing the design of complex systems," *IEEE Transactions on Engineering Management*, Vol. EM-28, No. 3, 1981, pp. 71–74.

[23]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.

[24]Holten, D., "Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 12, No. 5, 2006, pp. 741–748.

[25]Bostock, M., "D3.js website: `https://d3js.org/`," 2015.

[26]van Gent, I., Aigner, B., La Rocca, G., Stumpf, E., and Veldhuis, L. L. M., "Using graph-based algorithms and data-driven documents for formulation and visualization of large MDO systems," *6th CEAS Air and Space Conference*, 2017.

[27]Jepsen, J., Ciampa, P. D., and Nagel, B., "Avoiding Inconsistencies Between Data Models In Collaborative Aircraft Design Processes," *DLRK 2016*, 2016.

[28]Lefebvre, T., Bartoli, N., Dubreuil, S., Panzeri, M., Lombardi, R., Della Vecchia, P., Nicolosi, F., Ciampa, P. D., Anisimov, K., and Savelyev, A., "Methodological enhancements in MDO process investigated in the AGILE European project," *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2017.

[29]Jepsen, J., Ciampa, P. D., and Nagel, B., "Design of a Common Library to Simplify the Implementation of Aircraft Studies in CPACS," *DLRK 2015*, 2015.