From Requirements to Product: an MBSE Approach for the Digitalization of the Aircraft Design Process

Bruggeman, A.M.R.M.; la Rocca, G.

DOI
10.1002/iis2.13107

Publication date
2023

Document Version
Final published version

Published in
INCOSE International Symposium

33rd Annual INCOSE international symposium
hybrid event
Honolulu, HI, USA
July 15 - 20, 2023

From Requirements to Product: an MBSE Approach for the Digitalization of the Aircraft Design Process

Anne-Liza M.R.M. Bruggeman
Delft University of Technology
Faculty of Aerospace Engineering
Kluyverweg 1, 2629 HS, Delft
+31 15 278 5384
A.M.R.M.Bruggeman@tudelft.nl

Gianfranco La Rocca
Delft University of Technology
Faculty of Aerospace Engineering
Kluyverweg 1, 2629 HS, Delft
+31 15 278 5384
G.LaRocca@tudelft.nl

Abstract. During the aircraft conceptual design phase, many different design options need to be explored and compared in a short time frame. To speed up this process, efforts have been made in the past decades to digitalize parts of the design process, with a focus on the automation of the repetitive and non-creative tasks inherent to the iterative design process. Whilst many of the newly developed methodologies focus on specific parts of the design process, a holistic model-based design framework, incorporating the latest design technology developments, is lacking. To fill this gap, this paper presents the latest version of the Design and Engineering Engine (DEE) framework, originally proposed in the early 2000s and progressively matured through the experience of several international research collaborations. The DEE enables the setup and execution of Multidisciplinary Design Analysis and Optimization (MDAO) problems for aircraft (sub)systems, leveraging the automated, rule-based modeling capabilities offered by Knowledge-Based Engineering (KBE) and recent developments in the automatic formulation and integration of MDAO workflows. While the traditional MDAO process focuses on a given product architecture, the DEE allows also architectural design studies and makes use of Model-Based Systems Engineering (MBSE) principles to address the whole design process, from requirements modeling up to the automatic verification of the requirements. In practice, the DEE provides a single conceptual framework or template from which specific design framework instances can be formulated and executed, according to the user's needs. This paper describes the DEE architecture and its implementation concepts. Furthermore, it demonstrates the application of the DEE template to four different scenarios, ranging from a simple requirement verification study, up to the simultaneous synthesis and optimization of an aircraft system and its production process, including multiple system architecture options.

Nomenclature

CAD	=	Computer-Aided Design
CMDOWS	=	Common MDO Workflow Schema
DEE	=	Design and Engineering Engine
DoE	=	Design of Experiments
FEM	=	Finite Element Method
GUI	=	Graphical User Interface
KADMOS	=	Knowledge- and graph-based Agile Design for Multidisciplinary Optimization System

KBE	=	Knowledge-Based Engineering
MBSE	=	Model-Based Systems Engineering
MDA(O)	=	Multidisciplinary Design Analysis (and Optimization)
MMG	=	Multi Model Generator
PIDO	=	Process Integration & Design Optimization
RVF	=	Requirement Verification Framework
XDSM	=	eXtended Design Structure Matrix

Introduction

During the conceptual design phase of a complex product, such as a complete aircraft or aircraft subsystem, many different design options need to be explored and compared in a short time frame to make proper design choices. However, the number of designs that can be evaluated at early design stages is limited due to the manual, repetitive, and time-consuming tasks that are inherent to the aircraft design process, especially considering the trend of increasing number of disciplines (e.g., to address climate impact) and their fidelity level, already in conceptual design. Therefore, efforts have been made in the last decades to develop methodologies that digitalize and automate parts of the design process to reduce the development time and to enable the evaluation of more designs in early design stages.

One of these methodologies is Multidisciplinary Design Analysis and Optimization (MDAO), which brings together systems engineering principles and numerical optimization methods to connect different disciplines (e.g., aerodynamics, structures, production, etc.) and exploit their synergy to obtain an optimal design from a holistic point of view. With MDAO, one can set up design workflows that can automatically evaluate different design options against user-defined objectives and constraints. This increases the number of design evaluations that can be performed in a given time frame (Flager & Haymaker 2007) and enables a more efficient exploration of the design space. More what-if studies can be performed, e.g. by setting up MDAO problems with different objectives, constraints or design variables, thereby allowing designers to make better informed design choices.

However, setting up an MDAO workflow manually is a time-consuming and error-prone task (Flager & Haymaker 2007). Furthermore, once the MDAO workflow has been properly set up, it may be difficult to adapt, for example, to add or replace disciplinary analyses or change the optimization problem formulation. Therefore, research has been performed into the automatic formulation and execution of MDAO workflows (Gallard et al. 2018; Hoogreef 2017; Page Risueño et al. 2020; van Gent & La Rocca 2019). Starting from a repository of tools, the full MDAO workflow can automatically be formalized. By storing the formalized workflow in a standard data schema (for example as proposed by (van Gent, La Rocca & Hoogreef 2018)), the workflow can be imported into a Process Integration and Design Optimization (PIDO) tool to automatically materialize and subsequently execute the workflow. This methodology makes the design process more agile and less error-prone, as new studies can be set up automatically and existing design problems can be reformulated faster without having to manually adapt the workflow.

Another methodology to accelerate the design process of complex products through the automation of repetitive and non-creative tasks is Knowledge-Based Engineering (KBE). KBE provides a powerful high-level programming language to formalize rule-based engineering design procedures and a tight integration to a CAD kernel for the generation and manipulation of geometries. Using KBE, designers can build software applications to automatically generate and flexibly configure complex system architectures and their related disciplinary models (views), directly tailored to the needs of external analysis tools. This can be achieved thanks to the so-called *generative design* capabilities of KBE systems, which can fully automate the generation and modification of the system of interest model, including its disciplinary views, all starting from a single data set, which can be edited by the designer or an optimizer (La Rocca 2012). By using KBE, the time to create a model of a specific

system architecture is drastically reduced with respect to traditional CAD work (La Rocca 2012), such that the evaluation of more design options in a given time frame as well as the exploitation of MDAO is enabled.

Lastly, Model-Based Systems Engineering (MBSE) is a methodology that focuses on the digitalization of the systems engineering process, where traditional document-based artifacts are replaced by interconnected digital models. In a typical design problem, thousands of requirements need to be taken into account. These requirements continuously change due to changes in design specifications from the customer or changing markets (Mavris & Pinon 2012; Pernstål, Magazinius & Gorschek 2012). By using models to represent the system and the requirements, connections between the different elements are made explicit, making the system more transparent and facilitating change management. Furthermore, MBSE can be used to take the requirements automatically into account in the design process, enabling the formulation of MDAO systems specifically defined to seek requirement-compliant designs, thus enabling true *design by requirements* (Bruggeman et al. 2022).

While all the methodologies mentioned above contribute to the digitalization and automation of the design process, each of them focuses on different parts of this process. Apart from the initiatives addressed below in this paper, little research has been performed in combining these methodologies into one holistic model-based design framework, which is scalable and suitable for collaborative design; i.e., enabling different stakeholders to contribute with their needs, tools and expertise to a design process systematically driven by requirement verification.

This paper addresses this gap by presenting the latest version of the Design and Engineering Engine (DEE). Originally proposed in the early 2000s by (La Rocca & van Tooren 2007; La Rocca & van Tooren 2009; Morris et al. 2004), the DEE is a *conceptual framework* continuously developed through the experience and lessons learned from several international research collaborations, such as MOB[1], IDEALISM[2], AGILE[3], AGILE 4.0[4] and the currently running DEFAINE[5] project. The DEE enables the generation of use case specific computational design systems for MDAO. In its latest version, it leverages recent developments in the automatic formulation and integration of MDAO workflows, makes use of MBSE principles to address the whole design process from requirements modeling up to their automatic verification, and includes the product architecting phase in the MDAO process.

The structure of the paper is as follows. First, an overview of the current version of the DEE is provided and a detailed explanation of its logical structure is given. Next, the current status of the DEE technical implementation is discussed. As the DEE aims to be a general framework, its application to a range of use cases is described, demonstrating its effectiveness to act as a blueprint for diverse design studies. Lastly, the originality and useability of the DEE is discussed.

The Design and Engineering Engine

A diagrammatic representation of the current Design and Engineering Engine concept is shown in Figure 1. The DEE is not a fixed computational framework, but a conceptual framework. Therefore, it should be intended as a blueprint to derive specific architectures of design systems, integrated according to the design case at hand, always starting from requirements modeling and finishing with the generation of a design solution and related compliance report. Different instances of the DEE are described later in this paper. Here below, a step-by-step description of the process shown in Figure 1 is provided.

[1] https://cordis.europa.eu/project/id/G4RD-CT-1999-00172, accessed on: 24-04-2023
[2] https://idealism.ifb.uni-stuttgart.de/, accessed on: 24-04-2023
[3] https://www.agile-project.eu/, accessed on: 24-04-2023
[4] https://www.agile4.eu/, accessed on: 24-04-2023
[5] https://www.defaine.eu/, accessed on: 24-04-2023

The process starts with the definition of the *top-level requirements* for the system of interest, e.g. an aircraft. A stakeholder analysis is performed and from their needs, requirements are derived. Requirements can be divided into several categories. The functional requirement category includes requirements that *'specify a function that a system or system component shall perform*' (International Organization for Standardization 2017). The functional requirements are used to derive the so-called *architectural design space* model.

The architectural design space consists of all the possible logical architectures the system of interest could have. It is modeled by connecting logical components to the functions derived from the functional requirements. One function can be fulfilled by multiple components. In this case, a design choice must be made, which leads to the definition of different system architectures. Once a component is introduced in the model, it usually induces new functions that must be fulfilled by other components. An example of an architectural design space is shown in Figure 2. More details and information on architecture modeling can be found in (Bussemaker & Ciampa 2022).

Once the architectural model is complete, *architectural design choices* must be made to obtain one *system architecture* instance. The architectural design choices can be made either by the designer or by a computer during the analysis or optimization of the logical system architecture (Bussemaker et al. 2021). An example of a selected system architecture is shown in green in Figure 2. As soon as a choice is made for a specific component, new induced requirements, also called *architecture specific requirements*, are introduced. For example, choosing an electrical engine will lead to different design requirements than choosing a gas turbine.

After all top-level and architecture specific requirements have been collected, verification methods need to be selected for all non-functional requirements. A verification method consists of a *means of compliance* and a *test case*. A means of compliance is defined as

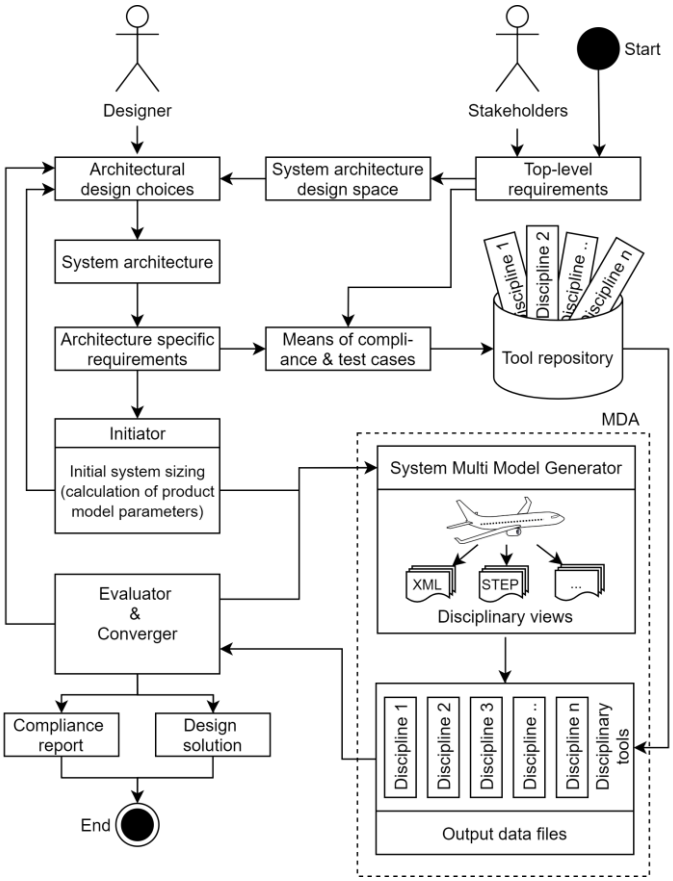

Figure 1: Overview of the Design and Engineering Engine framework (only main connections shown for readability)

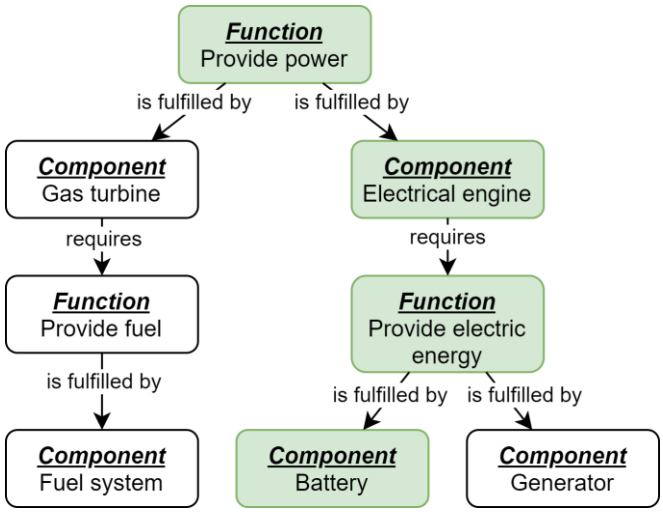

Figure 2: System Architecting example for an engine starting from the functional requirement '*The engine shall provide power*'. If multiple components can fulfill the same function, an architectural design choice has to be made. An example of a system architecture instance derived from this model is indicated in green.

follows: '*The agreement between the need stakeholder and the responsible stakeholder on how requirement compliance will be shown*' (Bruggeman et al. 2022). The need stakeholder is the one from whom the requirement is originating. The responsible stakeholder is the one who needs to show compliance with the requirement. The means of compliance is materialized in test cases that are used during the design process. A test case "*consists of all the* [system] *models, analysis tools, and physical tests (including the required input and output variables) that are necessary to verify the compliance of requirement given a fixed design*" (Bruggeman et al. 2022). Examples of means of compliance and test cases can be seen in Figure 3. More details and information on means of compliance and test cases can be found in (Bruggeman et al. 2022).

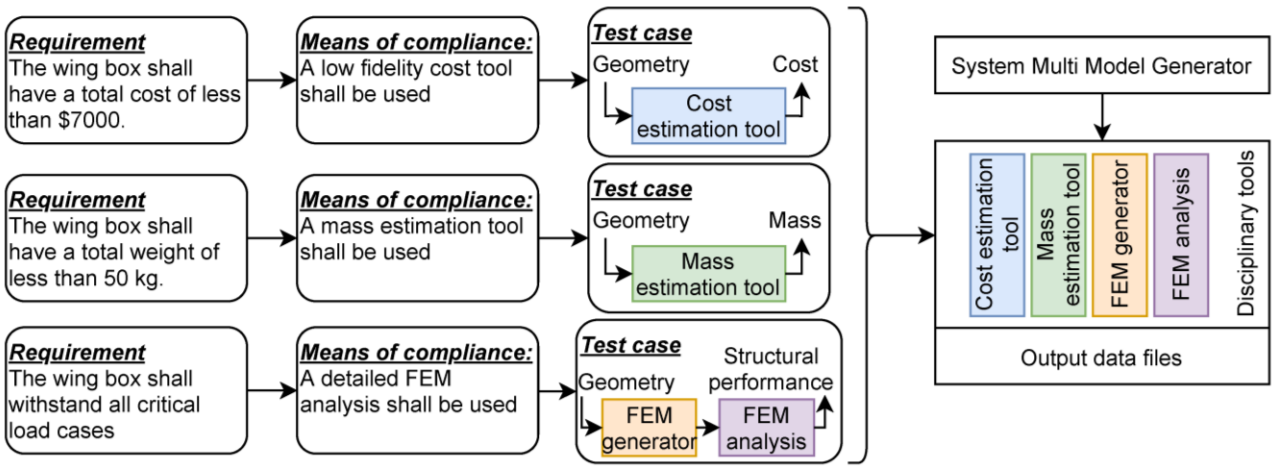

Figure 3: Example of how the means of compliance and test cases are connected to the requirements and how the disciplinary tools are used to evaluate the system.

When all the requirements and their verification methods are known, a first estimate of the size of the system needs to be made. In other words, a first estimation of the physical system architecture needs to be made. This is achieved by using the *Initiator* (Figure 1), whose input consists of a selected logical system architecture and its related requirements. The output is an (initial) set of values for the parameters of the product model addressed below. The role of the Initiator can be taken by a computational tool implementing conceptual/preliminary design rules, as the one described by (Elmendorp, Vos & La Rocca 2014), or directly by the designer, who proposes an initial product design, e.g. derived from a previous project. Sometimes, an initial design cannot be generated for a given architecture and set of requirements, in which case an iteration of the previous DEE steps is required. A different architecture may be selected or a request to relax some requirements can be proposed.

Based on the initial product parameters, a system model is created by the *System Multi Model Generator*. The System Multi Model Generator acts as the single source of truth from which the different disciplinary views, or abstractions, can automatically be derived to feed the multidisciplinary analysis process. Product views, such as the 3D geometrical representation of a wing outer surface, or the FEM model of its internal structure, are required in the next step as input for the *disciplinary tools*. The System Multi Model Generator can vary from a simple (analytically defined) parametric model to a sophisticated KBE model to enable CAD-in-the-loop MDAO (Sobieszczanski-Sobieski, Morris & Van Tooren 2015), as briefly mentioned in the Introduction section. In MDAO parlance, the combination of the System Multi Model Generator and the disciplinary tools is addressed as the Multidisciplinary Design Analysis (MDA) block of the design system (see dashed contour in Figure 1).

The tools fed by the System Multi Model Generator are those required for the system analysis and depend on the means of compliance and test cases agreed upon for the active set of requirements. As is shown in Figure 3, tools that are part of the test cases are collected and used to evaluate the system. A tool is thus only present in a DEE instance when needed to check or guarantee compliance with a

given requirement. Note that when physical tests or manual steps are part of the test cases, the design process cannot be fully automated and will require designer intervention.

The result of the system analysis is passed to the *Evaluator & Converger*. The form of the evaluator depends on the design study. When the goal is to optimize the design, the evaluator will take the form of an optimizer. The formulation of the optimization problem is strictly dependent on the system requirements, as each requirement is assigned a specific role in the MDAO problem formulation, e.g. objective, constraint, design variable, design variable bound, input parameter or quantity of interest (Bruggeman et al. 2022). Depending on the formulation of the MDAO problem, the optimizer can change the design point for a given logical system architecture (i.e. perturb the initial design produced by the Initiator), or even change the system architecture by making different architectural design choices. In the latter case, also the architecture specific requirements change. As the disciplinary tools are derived from the requirements, changing requirements can lead to different disciplinary tools in the MDA. This requires a very flexible System Multi Model Generator, as it must adjust the system model to the new system architecture and create different disciplinary views for the new set of disciplinary tools. Furthermore, it requires a very flexible workflow management system to reformulate the MDA and it requires specific optimization strategies to handle possible changes in the design variables and constraints. MDAO workflows in which the design variables, constraints, and disciplinary tools can vary based on the current design point are called *dynamic workflows*. Note that the use of the DEE is not restricted to optimization studies only. DEE instances can be assembled to formulate "simple" analysis workflows (MDAs) or perform parametric studies, e.g. based on some Design of Experiments (DoE) strategy. In the latter case, the evaluator block feeds the MDA with the different DoE points. A converger is present when multiple disciplinary tools are coupled and thus depend on each other.

Once the optimization, DoE or design analysis is completed, the obtained *design solution(s)* are evaluated against the set requirements. A *compliance report* is generated, which shows for each requirement whether compliance has been achieved and with what margin, as well as the analysis tools used to check the compliance (Bruggeman et al. 2022).

A key condition to enable the agile (re)configuration of the DEE, such that different and diverse design studies can be supported, is the adoption of a central data schema (Moerland et al. 2020; Nagel et al. 2012). All elements in the DEE need to exchange data with each other. To avoid having many specific interfaces between the different elements, a central data schema is used which enables a plug-and-play approach for any DEE component. As shown in Figure 4, each element in the framework only needs to have one interface with the central data schema and, through that, can immediately communicate with all other elements in the framework. Figure 4 also indicates what kind of data each element of the DEE exchanges with the central data schema. Note that also the disciplinary views generated by the System Multi Model Generator need to be translated to the central data schema. Transferring highly specific and complex models (for example a FEM model) to a central data schema might

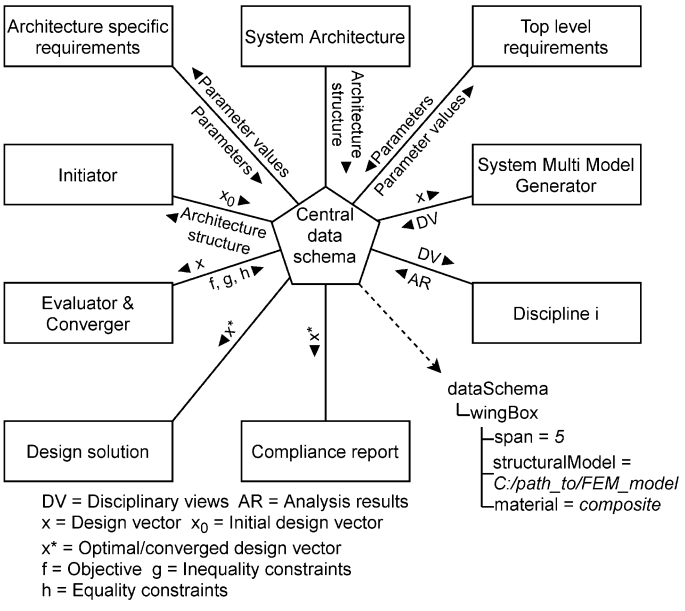

Figure 4: A central data schema is used to let the different elements of the DEE communicate with each other. The figure indicates what kind of information each element exchanges with the central data schema

be very difficult or even impossible. Therefore, a node in the central data schema can also point to a specific data set (e.g. a FEM model), stored outside the central data schema (Figure 4, bottom right). The central data schema can take any form, as long as all the DEE elements use the same schema.

Implementation Concepts

While the previous section provided an overview of the DEE conceptual framework, introducing all the components and steps involved in the overall design (and optimization) process, this section focuses on three of the implemented concepts and methodologies that effectively enable the digitalization and automation of such a process, namely: modeling of the requirements, modeling of the system architecture design space, and automatic formulation and execution of the MDAO workflows.

Requirements modeling

An important part of the DEE implementation is the modeling of the requirements and associated elements, such as stakeholders, needs, means of compliance, etc. Following the principles of MBSE, one model is created that acts as the single source of truth from which specific data and views can be extracted. This model is represented by a graph as shown in the center of Figure 5. A graph consists of nodes and edges. The nodes represent the different elements (e.g. stakeholders, requirements, disciplinary tools) and the edges represent the connections between the different elements (e.g. a need that originates from a stakeholder). The use of a graph representation is convenient to store, query and visualize data for inspection. Attached to this model are several modules that interact with the requirements graph.

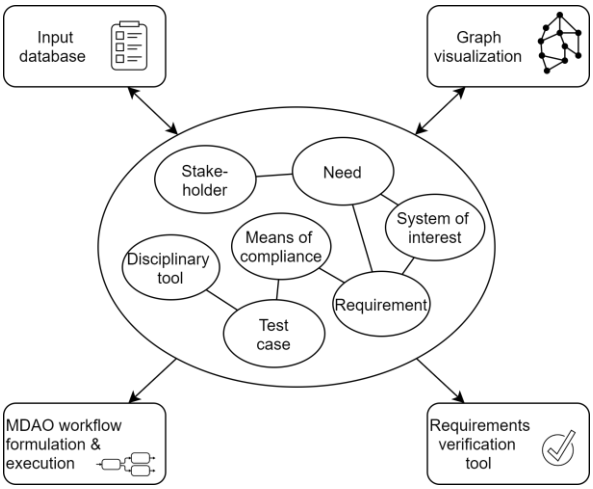

Figure 5: Overview of the requirements graph and the modules that interact with the graph

The first is the input module (Figure 5, top left). The input module is a GUI that is used to add information to the requirements graph. Within this GUI, the different elements, namely stakeholders, needs, system of interest, requirements, means of compliance, test cases and disciplinary tools are defined, together with their interrelationships. Furthermore, information related to the specific elements is given. For example, each requirement needs to be formulated according to a pattern to make it machine-readable. This means that the requirement's text is split up into small parts and each part is assigned a specific meaning which is computer interpretable. More information on requirements patterns and their utilization to automatically verify requirement compliance by means of a test case can be found in (Bruggeman et al. 2022). At the moment, the input module consists of a simple Excel spreadsheet, although any other software tool could be used.

The second module is the graph visualization module (Figure 5, top right). This consists of an interactive tool to inspect and visualize the requirement graph. The graph is exported into a GraphML schema (Brandes et al. 2010) and then imported into the software tool yEd[6] for visualization. Any other software tool compatible with GraphML could be used as well. The benefit of the visualization tool is that it allows the user to interactively navigate, and inspect the data attached to each element in the graph and the relations between the different elements. This improves traceability and transparency as one can see from which need and stakeholder a given requirement is originating and how compliance can be shown, by means of what test cases. One can also extract the list of all the design and analysis tools required for the design case at hand, where the presence of every single tool is

[6] https://www.yworks.com/products/yed, accessed on: 28-04-2023

justified by its inclusion in at least one test case. An example of the graph visualization is shown in Figure 6.

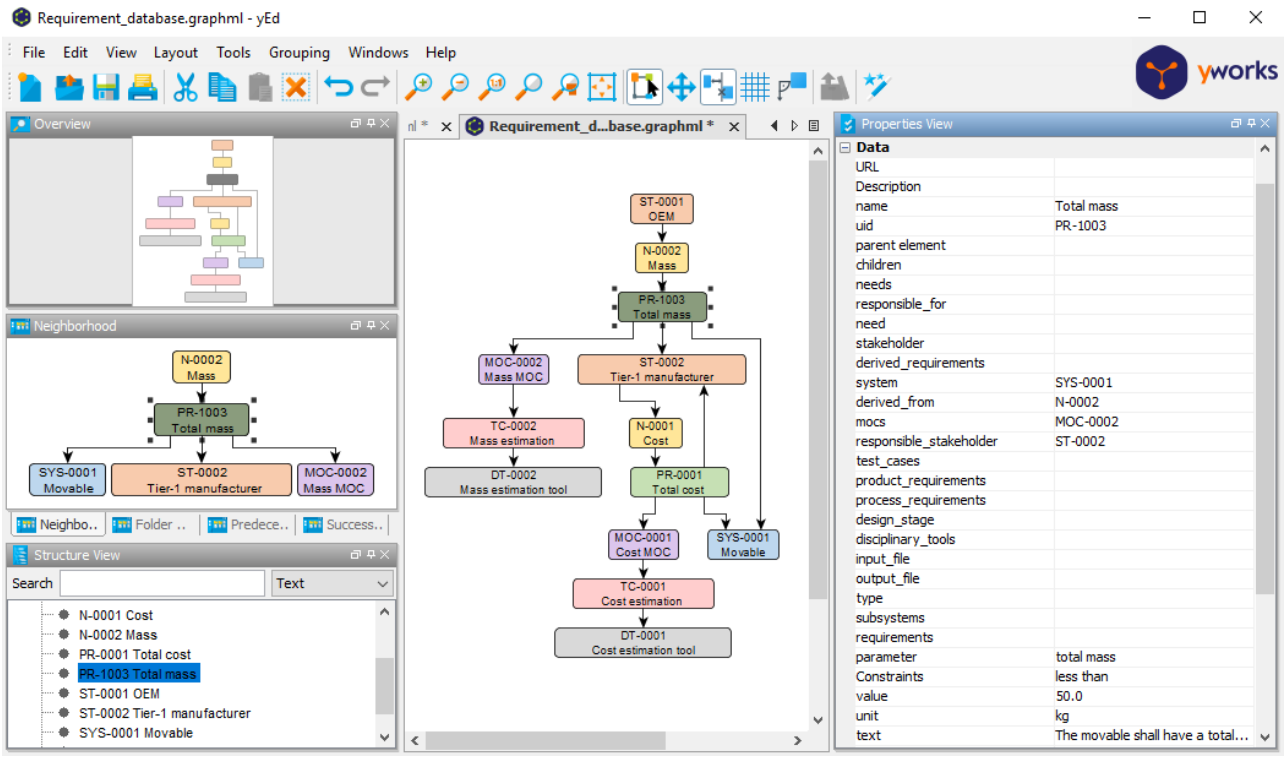

Figure 6: Example of a requirement graph visualization and inspection using yEd

The third module is the requirements verification tool (Figure 5, bottom right). The requirement verification tool is used to assess a specific system model against a set of requirements. To this purpose, the disciplinary tools needed to check requirement compliance are extracted from the requirements graph, integrated into a simulation workflow, and executed to evaluate the system. The actual values of the system are compared against the target values specified in the requirements and the compliance report is automatically generated. The compliance report (example in Figure 7) indicates whether requirements are met and with what margin (called 'Difference' in Figure 7). In the future, the report will also indicate the analysis tools involved in the compliance check.

	Requirements	Textual Requirements	Compliance	Value	Unit	Difference
R-1001	total cost	The movable shall have a total cost of less than $5000	False	5088.42	$	-1.77%
R-1004	total bracket cost	The movable shall have a total bracket cost of less than $2000	True	1562.48	$	21.88%
R-1003	total mass	The movable shall have a total mass of less than 50 kg.	True	19.38	kg	61.24%
R-2007	span	The movable shall have a span of less than 3000 mm.	False	3757.34	mm	-25.24%

Figure 7: Example of a compliance report (adapted from (Bruggeman et al. 2022))

The fourth, and possibly most distinctive module, is the one for automatic MDAO workflow formulation & execution (Figure 5, bottom left). This module provides capabilities well beyond the "simple" assessment of requirement compliance: it formulates and executes an MDAO workflow aimed at designing a system that meets its requirements at best. This module is explained in more detail below.

System architecture modeling

The system architecture can be modeled according to the techniques described, for example, in (Bussemaker & Ciampa 2022). By identifying the functions a system must fulfill, adding components to the functions, and determining the derived functions from the components, an architectural design space is built. Thereafter, individual system architectures must be extracted from this design space and fed to the subsequent sizing and evaluation processes in the DEE. Therefore, the system architecture model must be connected to the requirements model. This is achieved by adding a set of requirements to each component in the system architecture model. As soon as one architecture is selected, the requirements attached to each component of the given architecture become active as shown in Figure 8. A similar technique can be used to configure the System Multi Model Generator, where different building blocks (e.g. the High Level Primitives described in (La Rocca & van Tooren 2009)) are selected and combined to compose the full system model.

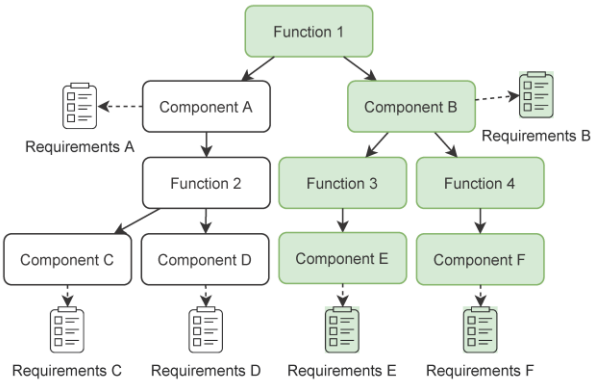

Figure 8: Example of connecting system architecting to the requirements model. The green blocks indicate the chosen system architecture

Automatic MDAO Workflow formulation & execution

Once the system model and requirements model are built, the design study needs to be set up and executed. This is achieved by using MDAO. A key step towards the formulation of the MDAO workflow consists of the assignment of "MDAO roles" to the different requirements by the user. MDAO role options are constraint, objective, design variable, design variable bound, input parameter and quantity of interest. Next, the disciplinary tools are collected from the requirements' test cases and connected, and the design variables, constraints, and objectives are formulated according to the roles assigned to each requirement. This formulation process of the MDAO workflows based on requirement roles, means of compliance, and test cases is fully automated in the DEE.

In the current DEE implementation, the automatic formulation of the MDAO problem is provided by KADMOS (van Gent & La Rocca 2019). KADMOS (Knowledge- and graph-based Agile Design for Multidisciplinary Optimization System) is a graph manipulation system that makes use of the aforementioned central data schema to connect all the tools in the repository into a user-selected MDAO architecture. Once the MDAO problem has been formulated, it needs to be translated into an

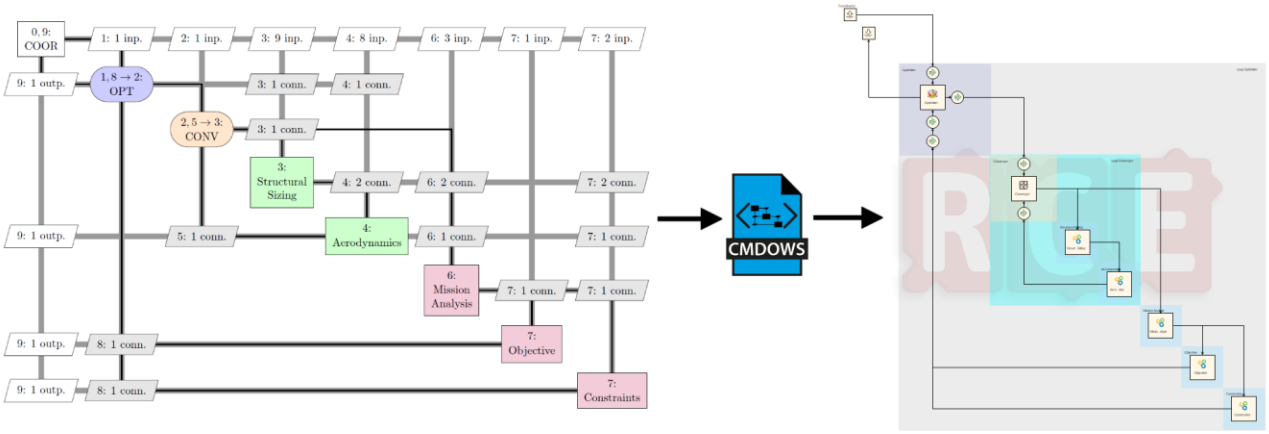

Figure 9: Example of an MDAO workflow automatically generated using KADMOS and translated into an executable RCE workflow using CMDOWS

executable workflow. This is achieved by using CMDOWS (Common MDO Workflow Schema) which is a proposed standard format to store and communicate MDAO workflows (van Gent, La Rocca & Hoogreef 2018). The CMDOWS files generated by KADMOS can then be imported into a Process Integration & Design Optimization (PIDO) tool, such as RCE[7], OpenMDAO (Gray et al. 2019) or Optimus[8], by means of dedicated translators. The complete executable workflow is then automatically built (materialized), ready for execution. An example of a formulated workflow, formalized by means of an automatically generated XDSM representation (Lambe & Martins 2012), and its executable counterpart integrated into RCE is shown in Figure 9.

Application of the Design and Engineering Engine

The DEE is a blueprint enabling the instantiation of different computational design systems, according to the design case at hand. In this section, examples of possible instantiations are provided for different use cases to demonstrate the flexibility of the proposed framework. The use case characteristics are summarized in Table 1 and further explained in the following sections. Note that these use cases are fictitious and show how the DEE can be applied in each of these scenarios.

Table 1: Overview of the different characteristics of four DEE use cases

	Use case 1: Compliance check	Use case 2: Aircraft synthesis	Use case 3: Aircraft synthesis and optimization	Use case 4: Simultaneous design optimization of product and production system
Architectural modeling	No	No	Yes	Yes
Architectural modeling choices made by	-	-	Designer	Optimizer
Initiator capability provided by	Designer	Designer	Empirical relations	Empirical relations
Type of System Multi Model Generator	CAD model	Analytical parametric model	KBE model	System: KBE model Production: Analytical parametric model
Type of disciplinary tools	Analysis tools	Analysis and sizing tools	Analysis, sizing, and design optimization tools	Analysis, sizing, and design optimization tools
Evaluator & Converger	None	Converger	Optimizer & Converger	Optimizer & Converger
Dynamic workflow	No	No	No	Yes

[7] https://rcenvironment.de/, accessed on: 21-04-2023
[8] https://www.noesissolutions.com/our-products/optimus, accessed on: 21-04-2023

Use Case 1: Requirements Compliance Check

The first use case entails the compliance check of a given aircraft design. In this case, the architecture of the system of interest (the aircraft) is fixed and evaluated against the set requirements to check its compliance. The DEE instance for this use case is shown in Figure 10. Both top-level and architecture specific requirements are present. As the system architecture is fixed, the generation of the architectural design space and related selection process are out of scope. The Initiator capability is provided by the designer who is responsible for setting the values of the system parameters required by the System Multi Model Generator, which is represented by a CAD model (see the corresponding block in the XDSM, Figure 14). The goal of this use case is to check requirement compliance, therefore only analysis tools are allowed in the MDA and no sizing or design tools, as they would alter the system. Furthermore, there is no need for an evaluator or converger. The compliance check block in Figure 14 compares the values computed by the MDA analysis tools with the values specified in the requirements and generates the system compliance report.

Use Case 2: Aircraft Synthesis

The second use case focuses on the synthesis of a given aircraft configuration, starting from a set of top level aircraft requirements. This means that, just as in the first use case, the aircraft architecture is fixed and the architectural design space and design choices blocks are not present in this DEE instance (Figure 11). The Initiator function is again provided by the designer who sets the initial values for the system parameters (e.g. based on an existing reference aircraft). The System Multi Model Generator is in this case represented by an analytical parametric model, i.e. a collection of data and equations that describe the system shape and dimensions, without producing the actual geometrical models, like in CAD. Therefore, there is no System Multi Model Generator block present in the XDSM shown in Figure 15. Even though the system architecture is fixed, both analysis and sizing tools are present in the MDA. A converger is present to manage the couplings between the various disciplinary tools. The process ends with the compliance check and the generation of the compliance report.

Use Case 3: Aircraft Synthesis & Optimization

The third use case is about the conceptual design and optimization of an aircraft. The conceptual design stage is characterized by analyzing different system architectures and performing trade-offs to make proper design decisions. As shown in Figure 12, a system architecture design space model is built and the designer is responsible for making architectural design choices. The Initiator is in this case represented by a set of semi-empirical relations (aircraft conceptual design textbook methods) to produce an initial system design. If the Initiator cannot find a feasible design solution, the designer may have to revise his/her architectural design choices (or renegotiate the set of top level aircraft requirements). For this use case, the aircraft system is modeled by means of a KBE application called the MMG (Multi Model Generator), which needs to be sufficiently flexible to model many different system architectures and, for each one, to generate the system views for the disciplinary tools involved in the MDA. In this case, both analysis, sizing, and design optimization tools are present in the DEE instance. The optimizer takes care of optimizing the given system architecture while guaranteeing compliance with the various requirements. Again, once the optimizer has reached convergence, a compliance report is generated. As shown in the XDSM in Figure 16, the data for the compliance check can come from the coordinator, optimizer, disciplinary tools, objective and/or constraints. This is because requirements can be taken into account differently in the optimization, e.g. as design variable bounds, constraints or input parameters. Note that in this DEE instance, one system architecture is sized and evaluated at the time, whereas the entire DEE workflow can be executed multiple times for different system architectures, enabling multiple trade-offs. Alternatively, one could configure a DEE instance where the optimizer is also responsible for the architecture design choices, as discussed in the next use case.

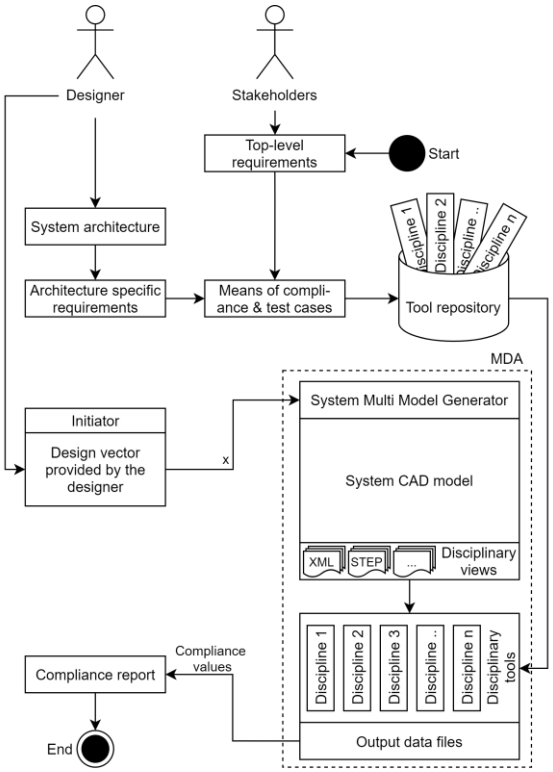

Figure 10: Overview of the DEE for Use Case 1: compliance check

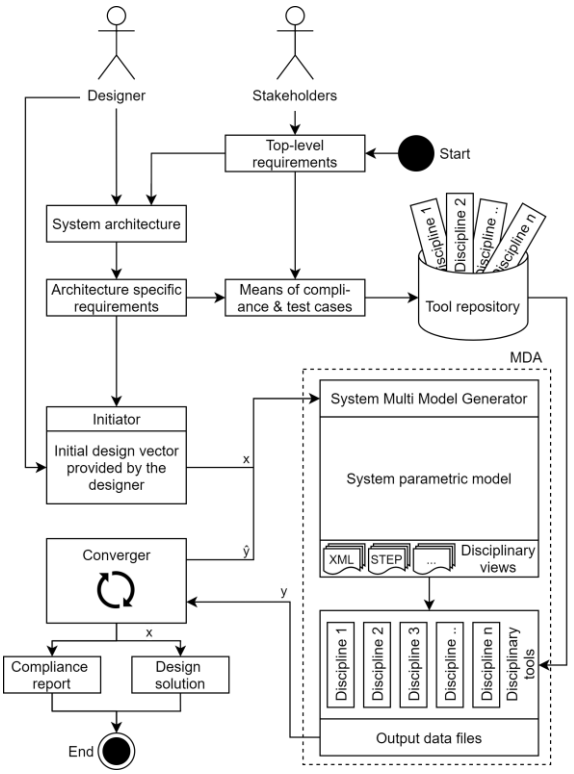

Figure 11: Overview of the DEE for Use Case 2: aircraft synthesis

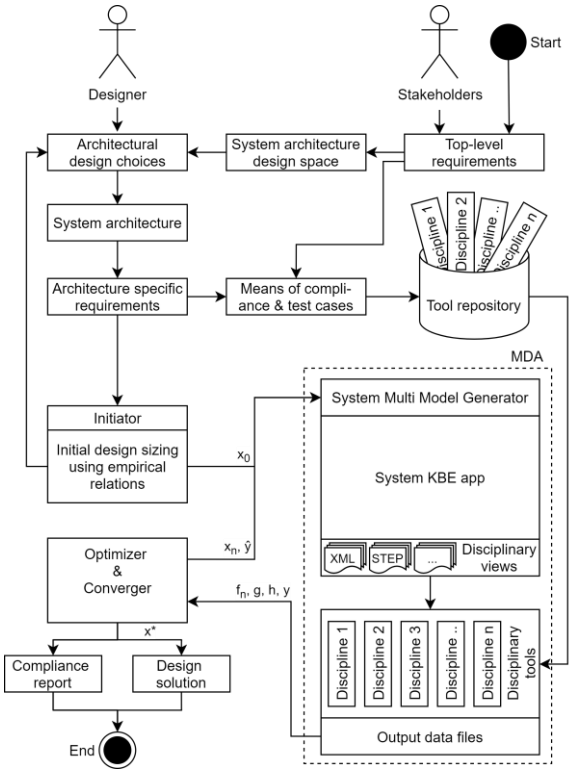

Figure 12: Overview of the DEE for Use Case 3: aircraft synthesis and optimization

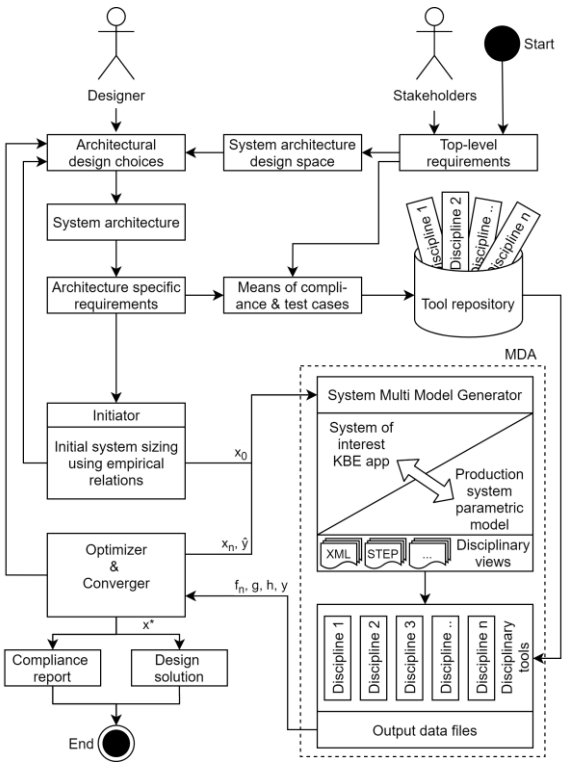

Figure 13: Overview of the DEE for Use Case 4: simultaneous aircraft and production system design & optimization

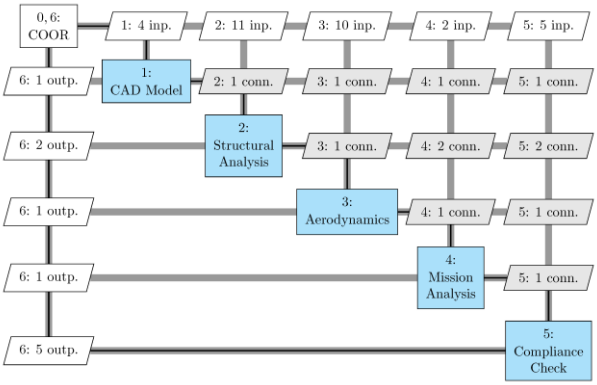

Figure 14: XDSM of the DEE instance for Use Case 1: compliance check

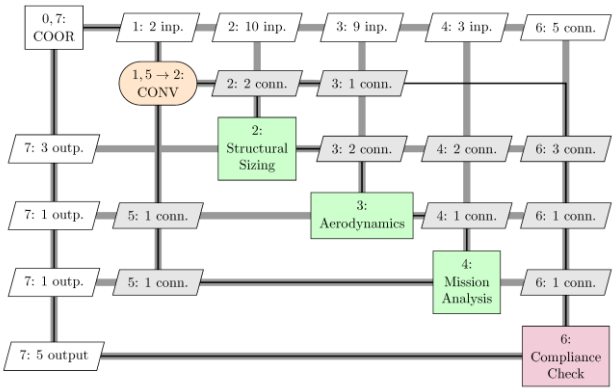

Figure 15: XDSM of the DEE instance for Use Case 2: aircraft synthesis

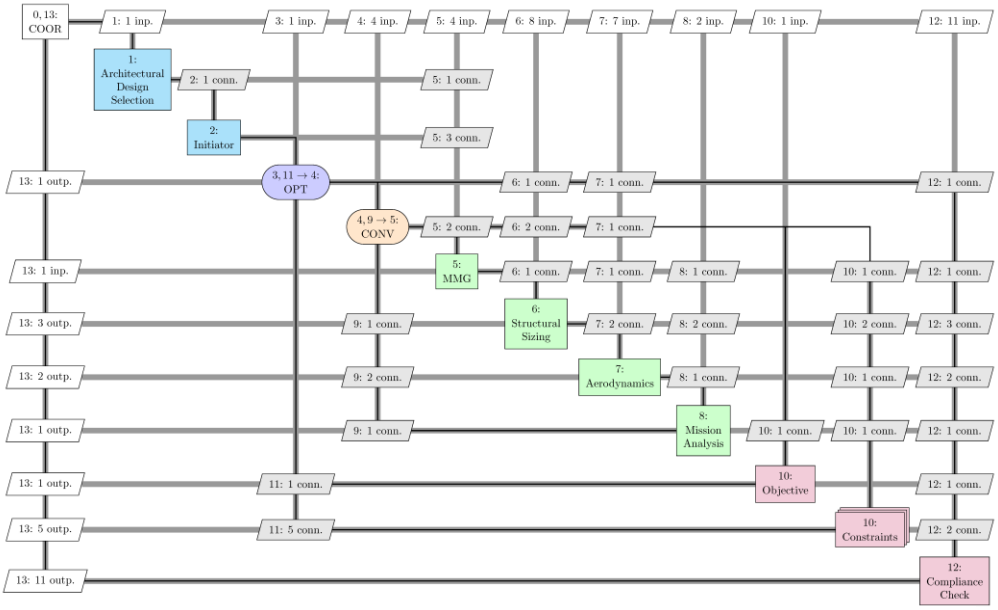

Figure 16: XDSM of the DEE instance for Use Case 3: aircraft synthesis and optimization

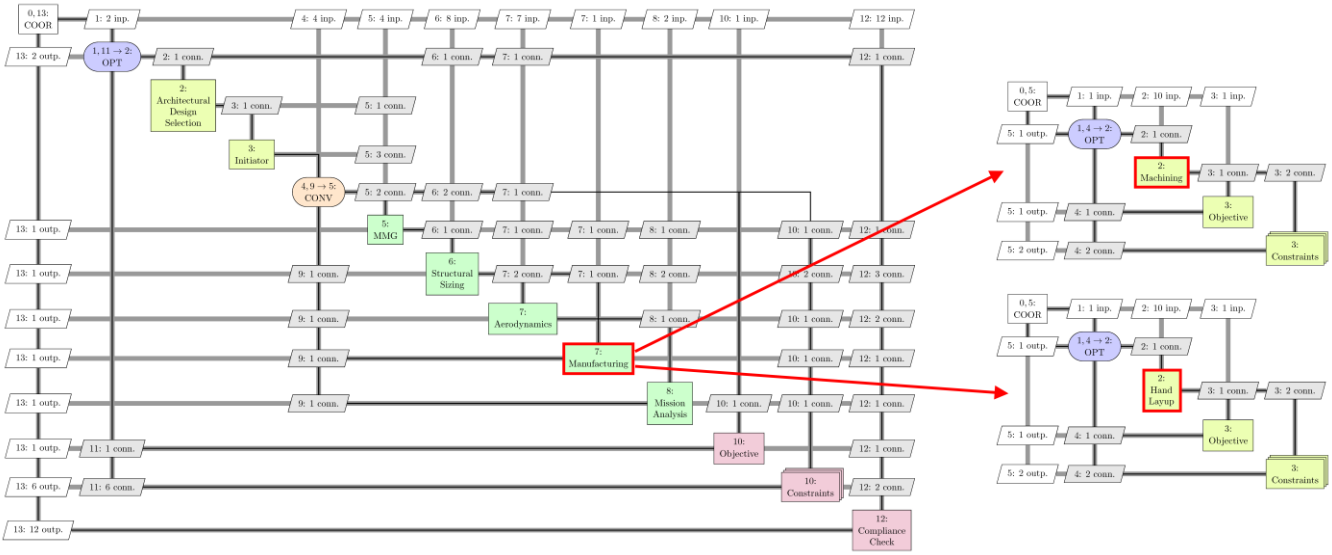

Figure 17: XDSM of the DEE instance for Use Case 4: simultaneous aircraft and production system design & optimization. The manufacturing block in the left XDSM is a sub-workflow, whose lay-out depends on the material choice: machining for metal (top right XDSM); hand layup for composite (bottom right XDSM)

Use case 4: Simultaneous Aircraft and Production System Design & Optimization

The last use case focuses on the simultaneous design of an aircraft and its production system. In this case, multiple systems interact with each other. Therefore, two System Multi Model Generators are present as shown in Figure 13. In this case, the aircraft is modeled using a KBE model (see MMG block in Figure 17), while the production system is modeled using an analytical parametric model. For both systems, a system architecture design space model is built and architectural design choices are made. The starting point for the system architecture is chosen by the designer. However, during the optimization, the optimizer is also allowed to make architectural design choices. Note that when a different architectural design choice is made, different architecture specific requirements become active, therefore different disciplinary tools will appear in the MDA. This means that the MDA needs to dynamically adapt according to the design point proposed by the optimizer. The way this dynamic workflow reconfiguration is tackled is shown in Figure 17. In this example, the manufacturing block represents a sub-workflow, whose layout depends on the material choice. If the design variable specifying the structure material is set to metal, the machining sub-workflow is executed to evaluate the manufacturing performance. When the design variable is set to composite by the optimizer, the hand layup sub-workflow is triggered. Each sub-workflow includes the design and analysis tools specific to the selected manufacturing approach, as required to guarantee compliance with the given manufacturing approach requirements.

Originality

Earlier publications have presented some of the concepts that have been implemented in the DEE (e.g. (Bruggeman et al. 2022; van Gent & La Rocca 2019)). Furthermore, many tools, both commercial as well as research tools, exist that can support the technical implementation of the DEE. Table 2 shows a non-exhaustive list of such tools, indicating which functionalities of the DEE they could support. However, as becomes clear from Table 2, none of the tools support all functionalities provided by the DEE.

The DEE provides a bigger framework that explains how the functionalities of these tools can be combined and how they complement each other, independent of the specific tools that are being used. The DEE combines MBSE, MDAO and KBE, leveraging the benefits of each methodology. For example, by using MBSE to model the requirements and their verification methods, the repository of tools required to formulate the MDAO workflow can be established. Furthermore, by combining the High Level Primitives of KBE with the system architecting process using MBSE, one can more easily obtain the system model that is required as an input for the MDAO process.

Another novelty of the DEE is that it explains how different systems of interest can be simultaneously designed while keeping track of the different requirements that each system of interest puts on the other system. For example, the current state of practice is that the producibility of a system is evaluated after the product design has been made. As shown previously in use case 4, the DEE indicates how the product and production process can be designed simultaneously by using different requirement databases, which automatically become active or inactive based on the current design choices. This increases the flexibility as one can more easily perform what-if scenarios with different production processes while staying compliant with the different production specific requirements. Furthermore, it increases the transparency and traceability of the design process as it becomes clear why certain requirements have been taken into account and how.

Table 2: Non-exhaustive list of commercial and research tools that can support functionalities of the DEE (to the best of the authors' knowledge)

	Requirement Management	Automatic Requirement Verification	System Architecting	MDAO Workflow Formulation	MDAO Workflow Execution
Commercial Tools					
IBM Doors[9]	✓	-	–	–	–
Jama Connect[10]	✓	✓	–	–	–
Dassault Systèmes (3DExperience[11])	✓	✓	–	–	✓
Siemens (Polarion[12] + Heeds[13])	✓	–	–	–	✓
Noesis Solutions (Optimus[14])	–	–	–	–	✓
Research Tools					
RVF (Bruggeman et al. 2022)	✓	✓	–	–	–
CCM (Raudberget, Edholm & Andersson 2012)	–	–	✓	–	–
ADORE (Bussemaker, Boggero & Ciampa 2022)	–	–	✓	–	–
KADMOS (van Gent & La Rocca 2019)	–	–	–	✓	–
InFoRMA (Hoogreef 2017)	–	–	–	✓	–
MDAx (Page Risueño et al. 2020)	–	–	–	✓	–
Gemseo (Gallard et al. 2018)	–	–	–	✓	✓
OpenMDAO (Gray et al. 2019)	–	–	–	–	✓
RCE (Boden et al. 2021)	–	–	–	–	✓
DEE	✓ RVF	✓ RVF	✓ TBD[15]	✓ KADMOS	✓ Optimus

[9] https://www.ibm.com/docs/en/ermd/9.7.0?topic=overview-doors, accessed on: 21-04-2023
[10] https://www.jamasoftware.com/platform/jama-connect/, accessed on: 21-04-2023
[11] https://www.3ds.com/3dexperience, accessed on: 21-04-2023
[12] https://polarion.plm.automation.siemens.com/, accessed on: 21-04-2023
[13] https://plm.sw.siemens.com/en-US/simcenter/integration-solutions/heeds/, accessed on: 21-04-2023
[14] https://www.noesissolutions.com/our-products/optimus, accessed on: 21-04-2023
[15] The requirements for the system architecture functionality of the DEE have been identified. However, no choice has yet been made with regard to the tool to be used.

Useability of the Design and Engineering Engine

An implementation of the DEE is currently under development at the Delft University of Technology. While some parts are already available, others are still under development, like the dynamic workflows and system architecting part. The bottom row of Table 2 indicates the tools that are currently being used for the implementation of the DEE. Except for Optimus, all tools are being developed by the Delft University of Technology. For the automatic formulation and execution of workflows, KADMOS, CMDOWS and Optimus are being used. KADMOS[16] and CMDOWS[17] are freely available and open-source. The Requirements Verification Framework (RVF) is being developed in collaboration with GKN Fokker and is therefore not freely available.

Making an explicit quantification of the benefits and useability of the DEE is challenging as it depends on the use case to which it is applied. However, earlier research performed by (Hoogreef 2017; Kulkarni, Hoogreef & La Rocca 2017) showed that the automated formulation of MDAO workflows was 90-97% faster than setting up the same workflows manually for several use cases. Other research has found a 40% time reduction for the formulation and integration of collaborative MDAO workflows (Ciampa et al. 2019). Furthermore, previous research found that the main benefits of automation and digitalization are the reduction of errors made during the setup of the design problem (Kulkarni, Hoogreef & La Rocca 2017). Application of the RVF as described by (Bruggeman et al. 2022) showed improved traceability from requirements to product design, as it shows how the design problem was formulated and what tools were used to achieve requirement compliance. Based on these results, similar benefits and results can be expected for the DEE.

The DEE makes use of fixed templates and patterns with which all elements must comply. For example, all disciplinary tools need to communicate with each other through the central data schema, all requirements need to be formulated according to specific patterns and all requirements need to be verified either through derived requirements or through means of compliance and test cases. Therefore, the expectation is that the effort to set up an initial DEE instance will increase compared to the current state of the art, as similar results were found when MDAO was introduced in the design process (Flager & Haymaker 2007). Furthermore, due to the many concepts and tools involved in the DEE, there will be a big learning curve and it will require a change in mindset. However, once the entire framework has been set up, it will be easier to perform a wide variety of design studies. Moreover, once certain elements (e.g. requirements or disciplinary tools) are formulated according to the fixed templates and patterns, they can easily be reused in different design studies.

Conclusions and Outlook

A flexible, transparent, traceable, and fast design process is crucial in the conceptual aircraft design phase to compare different design options and to come up with the best design according to the stakeholders' needs. For the past few decades, research has focused on the digitalization and automation of the design process, specifically focusing on the repetitive and time-consuming tasks inherent to the aircraft design process. However, these developments focus on specific parts of the design process. At present, a framework that combines all these developments is lacking.

The Design and Engineering Engine framework discussed in this paper aims at filling this gap by providing a flexible, reconfigurable conceptual framework or template to cover the entire design process, starting from the definition and formulation of requirements, through the system architecting, design, and optimization, until the automatic verification of requirements. The first version of the DEE was presented 20 years ago and since then has continuously evolved through the experience acquired in several international collaboration projects. In its current state, the DEE combines the

[16] https://gitlab.tudelft.nl/lr-fpp-mdo/kadmos, accessed on: 28-04-2023
[17] https://gitlab.tudelft.nl/lr-fpp-mdo/cmdows, accessed on: 28-04-2023

principles of MBSE with the design automation capabilities offered by KBE and MDAO. By modeling the requirements and their verification methods using graph theory, not only a single source of truth is created, but also multiple visualizations can be created, increasing the transparency and traceability of the requirements. From the requirements, complex MDAO workflows can automatically be generated, using the latest developments in automatic MDAO workflow formulation and integration. By combining the system architecting process with KBE, robust configurable system models can be created just starting from a list of parameters and multiple disciplinary views can automatically be generated, thus enabling CAD-in-the-loop MDAO. Furthermore, the DEE is scalable and suitable for collaborative design as multiple experts can contribute with their own tools due to the plug-and-play methodology, without having to install all tools on one machine.

Many different types of design studies can be formulated using different instantiations of the same DEE template. By removing or adapting elements, very different design cases can be addressed, ranging from simple requirement verification to multidisciplinary optimization including design architecture changes. Four exemplary use cases were described in this paper to show the flexibility and generality of the proposed design framework.

A full technical implementation of the DEE is under development. The next developments are focusing on the dynamic workflow capabilities addressed in the 4th use case in this paper. The objective is to let the MDAO workflow change based on the design point at hand: both disciplinary tools, as well as design variables, constraints, and objectives, can change if the system architecture is changed. At the moment, no consolidated methodologies exist to adapt the MDAO workflow during execution. Furthermore, the connection of dynamic workflows with system architecting will be developed, focusing on the activation of architecture specific requirements based on architectural design choices.

Acknowledgments

The research presented in this paper has been performed in the framework of the AGILE4.0 (Towards Cyber-physical Collaborative Aircraft Development) and DEFAINE (Design Exploration Framework based on AI for froNt-loaded Engineering) projects and has received funding from the European Union programs Horizon 2020 (grant agreement n° 815122) and ITEA 3 Call 6 project 19009.

References

Boden, B, Flink, J, Först, N, Mischke, R, Schaffert, K, Weinert, A, Wohlan, A & Schreiber, A 2021, 'RCE: An Integration Environment for Engineering and Science', *SoftwareX*, vol. 15,

Brandes, U, Eiglsperger, M, Lerner, J & Pich, C 2010, 'Graph Markup Language (GraphML)', in *Handbook of graph drawing and visualization*, Chapman & Hall.

Bruggeman, AMRM, van Manen, B, van der Laan, T, van den Berg, T & La Rocca, G 2022, 'An MBSE-Based Requirement Verification Framework to support the MDAO process,' *AIAA Aviation 2022 Forum*,

Bussemaker, J, Boggero, L & Ciampa, PD 2022, 'From System Architecting to System Design and Optimization: A Link Between MBSE and MDAO,' *32nd Annual INCOSE International Symposium*, 343-359.

Bussemaker, JH & Ciampa, PD 2022, 'MBSE in Architecture Design Space Exploration', in *Handbook of Model-Based Systems Engineering*, pp. 1-41.

Bussemaker, JH, De Smedt, T, La Rocca, G, Ciampa, PD & Nagel, B 2021, 'System Architecture Optimization: An Open Source Multidisciplinary Aircraft Jet Engine Architecting Problem,' *Aiaa Aviation 2021 Forum*,

Ciampa, PD, Prakasha, PS, Torrigiani, F, Walther, J-N, Lefebvre, T, Bartoli, N, Timmermans, H, Della Vecchia, P, Stingo, L, Rajpal, D, van Gent, I, La Rocca, G, Fioriti, M, Cerino, G, Maierl, R, Charbonnier, D, Jungo, A, Aigner, B, Anisimov, K, Mirzoyan, A & Voskuijl, M 2019, 'Streamlining Cross-Organizational Aircraft Development: Results from the AGILE Project,' *AIAA Aviation 2019 Forum*,

Elmendorp, R, Vos, R & La Rocca, G 2014, 'A Conceptual Design and Analysis Method for Conventional and Unconventional Airplanes,' *29th Congress of the International Council of the Aeronautical Sciences (ICAS)*,

Flager, F & Haymaker, J 2007, 'A Comparison of Multidisciplinary Design, Analysis and Optimization Processes in the Building Construction and Aerospace Industries,' *24th International Conference on Information Technology in Construction*

Gallard, F, Vanaret, C, Guénot, D, Gachelin, V, Lafage, R, Pauwels, B, Barjhoux, P-J & Gazaix, A 2018, 'GEMS: A Python Library for Automation of Multidisciplinary Design Optimization Process Generation,' *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*,

Gray, JS, Hwang, JT, Martins, JRRA, Moore, KT & Naylor, BA 2019, 'OpenMDAO: an open-source framework for multidisciplinary design, analysis, and optimization', *Structural and Multidisciplinary Optimization*, vol. 59, no. 4, pp. 1075-1104.

Hoogreef, MFM 2017, *Advise, Formalize and Integrate MDO Architectures*, PhD thesis, Delft University of Technology.

International Organization for Standardization 2017, *ISO/IEC/IEEE 24765 - Systems and software engineering - Vocabulary*,

Kulkarni, AR, Hoogreef, M & La Rocca, G 2017, 'Combining semantic web technologies and KBE to solve industrial MDO problems,' *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*,

La Rocca, G 2012, 'Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design', *Advanced Engineering Informatics*, vol. 26, no. 2, pp. 159-179.

La Rocca, G & van Tooren, MJL 2007, 'Enabling distributed multi-disciplinary design of complex products: a knowledge based engineering approach', *Journal of Design Research*, vol. 3, no. 5, pp. 333-352.

——— 2009, 'Knowledge-Based Engineering Approach to Support Aircraft Multidisciplinary Design and Optimization', *Journal of Aircraft*, vol. 46, no. 6, pp. 1875-1885.

Lambe, AB & Martins, JRRA 2012, 'Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes', *Structural and Multidisciplinary Optimization*, vol. 46, no. 2, pp. 273-284.

Mavris, D & Pinon, O 2012, 'An Overview of Design Challenges and Methods in Aerospace Engineering', in *Complex Systems Design & Management*, Springer, pp. 1-25.

Moerland, E, Ciampa, PD, Zur, S, Baalbergen, E, Noskov, N, D'Ippolito, R & Lombardi, R 2020, 'Collaborative Architecture supporting the next generation of MDAO within the AGILE paradigm', *Progress in Aerospace Sciences*, vol. 119,

Morris, A, Arendsen, P, La Rocca, G, Laban, M, Voss, R & Hönlinger, H 2004, 'MOB - A European Project On Multidisciplinary Design Optimisation,' *24th International Congress of the Aeronautical Sciences (ICAS)*,

Nagel, B, Böhnke, D, Gollnick, V, Schmollgruber, P, Rizzi, A, La Rocca, G & Alonso, JJ 2012, 'Communication in Aircraft Design: Can We Establish a Common Language?,' *28th International Congress of the Aeronautical Sciences (ICAS)*,

Page Risueño, A, Bussemaker, J, Ciampa, PD & Nagel, B 2020, 'MDAx: Agile Generation of Collaborative MDAO Workflows for Complex Systems,' *AIAA Aviation 2020 Forum*,

Pernstål, J, Magazinius, A & Gorschek, T 2012, 'A Study investigating Challenges in the Interface Between Product Development and Manufacturing in the Development of Software-Intensive Automotive Systems', *International Journal of Software Engineering and Knowledge Engineering*, vol. 22, no. 7, pp. 965-1004.

Raudberget, D, Edholm, P & Andersson, M 2012, 'Implementing the principles of Set-based Concurrent Engineering in Configurable Component Platforms,' *NordDesign 2012*,

Sobieszczanski-Sobieski, J, Morris, A & Van Tooren, MJL 2015, *Multidisciplinary design optimization supported by knowledge based engineering*, John Wiley & Sons.

van Gent, I & La Rocca, G 2019, 'Formulation and integration of MDAO systems for collaborative design: A graph-based methodological approach', *Aerospace Science and Technology*, vol. 90, pp. 410-433.

van Gent, I, La Rocca, G & Hoogreef, MFM 2018, 'CMDOWS: a proposed new standard to store and exchange MDO systems', *CEAS Aeronautical Journal*, vol. 9, no. 4, pp. 607-627.

Biography

Anne-Liza M.R.M. Bruggeman. Ir. A.M.R.M. (Anne-Liza) Bruggeman holds an MSc and BSc in Aerospace from the Delft University of Technology and is currently a PhD Candidate in the section of Flight Performance and Propulsion at the same university. Her doctoral research focuses on a new methodological approach to enable design for manufacturing in early design stages, using MBSE, MDAO and KBE.

Gianfranco La Rocca. Dr.ir. G. (Gianfranco) La Rocca holds an MSc in Aerospace Engineering from the University of Pisa, Italy and a PhD from Delft University of Technology in the Netherlands. He is associate professor in the section of Flight Performance and Propulsion of the same faculty where he pursued his doctoral title. He has developed and taught several BSc and MSc courses on Aircraft Design, Systems Engineering, and advanced design methodologies, such as KBE and MDAO. Dr. La Rocca's research activities focus on the development of design automation solutions to support the design of novel and more sustainable aircraft architectures and to contribute to the digitalization of complex, knowledge intensive design processes. As of 2000, Dr. La Rocca has been involved in many (inter)national projects related to aircraft design and collaborative MDAO, including the 2003 DESCARTES Prize finalist project MOB and the ICAS 2018 Innovation Award project AGILE.