AIAA AVIATION Forum
June 27-July 1, 2022, Chicago, IL & Virtual
AIAA AVIATION 2022 Forum

10.2514/6.2022-3722

An MBSE-Based Requirement Verification Framework to support the MDAO Process

Anne-Liza M.R.M. Bruggeman*
Delft University of Technology, Delft, The Netherlands

Bas van Manen†, Ton van der Laan‡, Tobie van den Berg§
GKN Aerospace, Papendrecht, The Netherlands

Gianfranco La Rocca¶
Delft University of Technology, Delft, The Netherlands

According to a study performed by the Project Management Institute, around 47% of unsuccessful projects do not meet their goals and objectives due to poor requirements management. Taking requirements into account during the aircraft design process and ensuring requirement compliance during all design phases is important to obtain good and feasible aircraft designs. However, a typical aircraft design process is very complex and many requirements need to be taken into account. This paper proposes a new framework that implements requirements in the design process by establishing a direct link between Model-Based Systems Engineering and Multidisciplinary Design Analysis and Optimization (MDAO). Model-based requirements are directly implemented in the optimization problem and based on the requirement verification methods the MDAO workflows are formulated. When requirements or verification methods change, the workflow is automatically updated accordingly. This way, requirement compliance can either be automatically enforced or checked based on the optimization or analysis results. Automatically generated requirement reports provide information on the requirement compliance results. The framework has been implemented in a software prototype, which was applied to the design of a wing box, showing the functionalities of the framework. With the framework, the traceability from requirements to product design is improved, as all stakeholders can see how the design process was formulated and how requirement compliance has been achieved. Furthermore, optimized designs can be obtained that satisfy all the stakeholders' needs.

Nomenclature

CMDOWS	Common MDO Workflow Schema
DoE	Design of Experiments
KADMOS	Knowledge- and graph-based Agile Design for Multidisciplinary Optimization System
KBE	Knowledge-Based Engineering
MBSE	Model-Based Systems Engineering
MDAO	Multidisciplinary Design Analysis & Optimization
MDA	Multidisciplinary Design Analysis
MDM	Multidisciplinary Modeller
PIDO	Process Integration and Design Optimization
RCE	Remote Component Environment
RVF	Requirement Verification Framework
SOTA	State of the Art

*PhD Candidate, Faculty of Aerospace Engineering, A.M.R.M.Bruggeman@tudelft.nl, AIAA Student Member
†Design Engineer, Centre of Competence Design, Fokker Aerostructures, Bas.vanManen@fokker.com
‡Manager, Centre of Competence Design, Fokker Aerostructures, Ton.vanderLaan@fokker.com, AIAA Member
§Engineering Specialist, Centre of Competence Design, Fokker Aerostructures, Tobie.vandenBerg@fokker.com, AIAA Member
¶Associate Professor, Faculty of Aerospace Engineering, G.LaRocca@tudelft.nl

I. Introduction

According to a study performed by the Project Management Institute, around 47% of unsuccessful projects do not meet their goals and objectives due to poor requirement management [1]. In product design, requirements must be well managed to ensure the final product meets the needs of all the stakeholders. This is especially important for aircraft design, where both the design process as well as the aircraft are becoming increasingly complex, making it more difficult to ensure all requirements are met. Good requirement management requires on the one hand a thorough understanding of the stakeholders' needs and a proper translation of the needs into requirements. On the other hand, it requires tracking, evaluating, and verifying requirements in the design process to ensure the final design is compliant with all the specified requirements.

One of the ongoing trends within requirements management is the switch from document-based to model-based requirements using Model-Based Systems Engineering (MBSE). Adopting a model-based approach has several benefits compared to a document-based approach. For example, using models enables the creation of a single source of truth from which all required information can be derived. This improves the consistency within the design process as information throughout the model is automatically updated. Another benefit of using a model-based approach is that models can be reused in different design projects, reducing the amount of rework required. Finally, the use of models improves transparency as the connections between the different models and elements can be inspected more easily.

Nowadays, requirements are often managed using dedicated software packages, like DOORS[1], Jama Connect[2], and the Requirements Manager in 3DExperience[3]. These software packages allow the user to model requirements and to connect them to system components and analysis methods to check for requirement compliance. However, these analysis methods are limited to a direct evaluation of the requirements given a pre-defined system design. What-if scenarios are evaluated by changing the system and analyzing the effect on the requirements. Extracts and reports from these tools are provided to the designers to serve as a (paper) guide during the design process. Requirements are not actively used to guide the design process, although they do limit the feasible design space during the design process as they act as constraints within the design problem. Therefore, it would be beneficial if requirements could be included directly in the design process to obtain optimal system designs that comply with all requirements.

When designing complex systems like aircraft, many disciplines are involved, each influencing the other. Multidisciplinary Design Analysis and Optimization (MDAO) accounts for these influences and exploits the synergy between different design aspects and disciplines to obtain an optimal design from a holistic point of view. An important hurdle in using MDAO has long been the difficult and lengthy setup time of MDAO problems [2]. However, recent developments [3–5] have made the setup of MDAO processes faster and more agile. Starting from a repository of tools, it is now possible to automatically make a formal formulation of the MDAO problem and then automatically translate this formulation into an executable workflow. The ability to quickly formulate MDAO workflows opens up possibilities to make a direct link between model-based requirements using MBSE and the design process using MDAO.

Previous research has been performed in making a direct link between MBSE and MDAO. Cencetti et al. [6] developed a framework that enables the evaluation of different design alternatives generated using MBSE in a fixed and manually constructed MDAO workflow. Jeyaraj et al. [7] developed a similar framework using Capella[4]. Capella models are enriched with variables, which are automatically implemented in the MDAO workflow as input values. The results from the MDAO execution are then automatically imported back into the Capella models. This enables the evaluation of many different design alternatives. Leserf et al. [8] used a different approach, as they enriched MBSE models with MDAO information. Parameters are marked as objectives in the MBSE models and design decisions are added through model variabilities. The enriched MBSE models are then translated into a Constraint Satisfaction Problem (CSP). Finally, Beernaert and Etman [9] used the Elephant Specification Language to transfer requirements into an Analytic Target Cascading (ATC) problem, enabling the optimization of different system levels. The main limitation of the frameworks described above is that they lack flexibility. The optimization processes are either fixed [6, 7] and manually constructed [6] or only specific optimization algorithms are supported like CSP [8] and ATC [9].

This paper proposes a new framework, called the Requirement Verification Framework (RVF), that establishes a more flexible link between MBSE and MDAO. The framework presents a new methodology that enables the derivation of MDAO problems from model-based requirements. Within the RVF, requirements are formulated in a machine-readable way and connected to the analysis methods that are required to verify the requirements' compliance. Based on the problem the user wants to solve (single design analysis, Design of Experiments (DoE), or optimization), different MDAO

[1] `https://www.ibm.com/products/requirements-management`, accessed on: 22-04-2022

[2] `https://www.jamasoftware.com/solutions/requirements-management/`, accessed on: 22-04-2022

[3] `https://www.3ds.com/3dexperience`, accessed on: 22-04-2022

[4] `https://www.eclipse.org/capella`, accessed on 27-04-2022

problem roles (e.g. design variable, objective, constraint, or quantity of interest) are assigned to the requirements by the user. The MDAO workflow can then automatically be derived, formulated, and executed. This supports the user in analyzing and optimizing system designs while ensuring the final design meets all the requirements and thus the stakeholders' needs. The automatic MDAO workflow generation allows users to easily reconfigure the design problem when stakeholders change, requirements are modified, or analysis methods are updated. Furthermore, as all elements in the process are modeled, a trace is established from the requirements to the design process, improving the *requirement traceability*[5]. A compliance report can automatically be generated, showing the user the requirements' compliance.

The structure of the paper is as follows. Section II explains the position of the RVF within the systems engineering process. The RVF consists of two parts. The first part focuses on the automatic verification of requirements and is described in Section III. The second part of the RVF connects the requirements to the MDAO workflow and is explained in Section IV. The entire framework has been implemented in a first software implementation. This prototype is described in Section V. Section VI shows the application of the prototype to the design of a wing box. Finally, Section VII draws some conclusions and discusses future work and recommendations.

II. The Requirement Verification Framework as Part of the Systems Engineering Process

The goal of the RVF is to support the user in integrating requirements in the design process, improve requirement traceability, and enable automatic verification of requirements. Using the RVF, different design options can automatically be designed and evaluated according to the same set of high-level system requirements, while simultaneously accounting for specific design-related requirements. This enables the evaluation of what-if scenarios and thereby supports the trade-off process to obtain the best designs according to the stakeholders' needs. The framework consists of the theoretical concepts and methods. A prototype implementation of the framework within a software package will be discussed in Section V.

The process that is followed within the RVF is shown in Figure 1 and consists of two parts. The first part (the top block in Figure 1) considers the automatic verification of requirements given a fixed design. The functionalities provided by this part are also provided by current SOTA requirement management tools. The second part (the bottom block in Figure 1) introduces new and innovative functionalities as it extends the process to include the connection between the requirements and the MDAO problem. In this part, the product design is iteratively and automatically changed such that the design complies with a given set of requirements, while it is simultaneously optimized for a given objective.

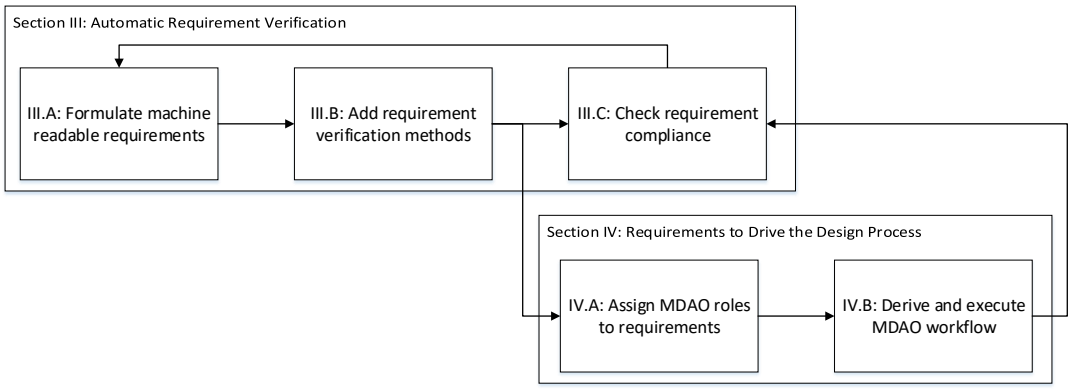

Fig. 1 Overview of the different steps within the Requirement Verification Framework. For each step, the dedicated paper (sub)section is indicated

The function of the RVF within a typical systems engineering process is shown in Figure 2 using the Vee Model (based on [11]). The parts supported by the RVF are indicated in bold. The systems engineering process starts in the top left with the generation of concepts. This includes the identification of the different stakeholders and their needs concerning the System of Interest and the creation of high-level concepts.

[5]*Requirement traceability* is defined here as: "*[a] discernible association between a requirement and related requirements, implementations, and verifications*" [10]. This means that there is a link (or trace) between the requirements, the system of interest, and the verification methods and that this link can be inspected by the user.

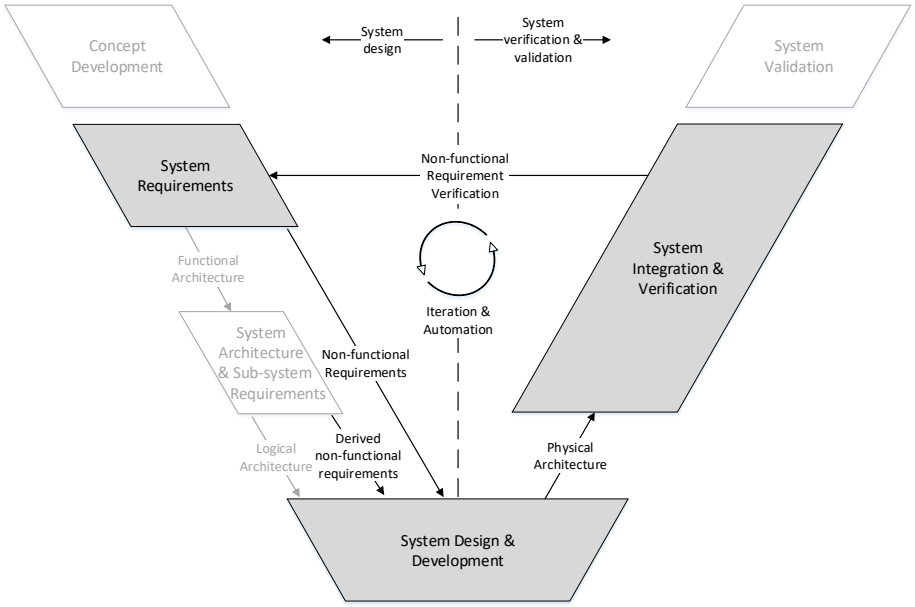

Fig. 2 The function of the Requirement Verification Framework within a typical systems engineering process using a Vee Model (based on [11]). The parts supported by the RVF are indicated in bold

The next step is to derive the system's requirements from the stakeholders' needs. The requirements can be split into two categories, namely functional and non-functional requirements. A functional requirement is a "*requirement that specifies a function that a system or system component shall perform*" [10]. Non-functional requirements are defined within this paper as all requirements that are not functional requirements. These include, amongst others, performance requirements, safety requirements, and quality requirements. Note that performance requirements differ from functional requirements. A performance requirement is defined as "*[a] measurable criterion that identifies a quality attribute of a function or how well a functional requirement shall be accomplished*" [10]. The RVF focuses on the quantitative analysis of requirements. As functional requirements indicate which functions a system should fulfill, they are not quantifiable. Therefore, only the non-functional requirements are considered within the framework. Once the system's requirements are identified, also the verification methods to check requirement compliance are defined and implemented in the RVF.

The functional requirements are used in the third step of the Vee Model to create several system architectures. Within the system architecting process, the different possibilities for subsystems and components are identified. With the introduction of each new subsystem or component, new derived requirements are introduced as well. These can again be divided into functional and non-functional requirements. The derived non-functional requirements are also covered by the RVF as shown in Figure 2. Note that the system architecting process is not part of the RVF. However, other frameworks exist that deal with this step in a model-based manner, for example, the framework developed by Bussemaker et al. [12].

In the lower part of the Vee Model, the full system, including its subsystems and components, is designed and developed. In this step, the RVF supports the user in setting up the MDAO problem. By letting the user assign MDAO problem roles (e.g design variable or constraint) to each requirement, a direct link is established between the requirements and the MDAO problem. Based on the verification methods that were specified for each requirement, the framework identifies which engineering tools and methods need to be implemented in the MDAO problem. As will be explained in more detail in Section IV, based on the requirements, the assigned MDAO problem roles, and the engineering tools, a formal formulation of the MDAO problem is automatically generated. The user can then inspect whether the formulation is correct or whether requirements and/or engineering tools are missing. Once the MDAO problem formulation is correct, it is automatically translated into an executable MDAO workflow. This process enables the optimization of the design according to the specified requirements. Besides optimization, it is also possible to generate and evaluate a DoE or single design analysis. This supports the user in evaluating what-if scenario's to make better design choices.

Once the system is designed, the requirements are automatically verified using the RVF and the verification methods that were specified earlier. The user can generate and inspect a requirement compliance report that indicates which requirements have been met and which have not. Based on these results, the user can choose to adapt the design in case of a DoE or single design analysis. In case an optimization was performed but no feasible solution was found, the requirements need to be reevaluated. The RVF supports the user in this process by automating several steps in the RVF process and supporting the iteration between the requirements definition, design process, and verification as is indicated in the middle of Figure 2.

The systems engineering process ends in the top right of the Vee model. In this step, the system is validated to check whether the final system design fulfills the stakeholders' needs. This step is out of scope for the RVF.

The RVF is completely model-based and object-oriented. This means that all elements, e.g. stakeholders, needs, requirements, system of interest, and verification methods, are objects within the framework and that connections exist between all these objects. This supports the user in implementing the correct requirements within the design process and maintaining requirement traceability. For example, if a stakeholder is removed, automatically the connected requirements are removed from the design process. Or if a subsystem or component is changed, its derived requirements are automatically updated. This ensures that one is using a complete and up-to-date set of requirements in the product development process.

III. Automatic Requirement Verification

The RVF consists of two parts. The first part focuses on the automatic verification of requirements based on a fixed design. As shown in Figure 1, this process consists of three steps. First, requirements must be formulated in a machine-readable manner. How this is achieved is explained in Section III.A. Next, Section III.B discusses how verification methods are defined and assigned to each requirement. In the last step, fixed designs are automatically verified based on the requirements and the specified verification methods. A compliance report is automatically generated, which can be inspected by the user. This step is explained in Section III.C.

A. Machine Readable Requirements

Requirement management starts with a proper definition of the stakeholders, needs, and requirements. Stakeholders express their needs and the needs are translated into requirements that can be verified. The methodology presented in this paper assumes that a thorough stakeholder analysis has already been performed and all the stakeholders' needs and requirements have been defined.

Once the requirements have been identified, they need to be formulated according to fixed patterns. The use of patterns makes the requirements machine-readable, as meaning is given to the different elements of the requirements. Figure 3 shows the patterns of the four requirement types that have been implemented within the RVF and that are originating from Carson [14]. The performance requirement indicates how well a function should be performed. The design constraint puts a limit on the feasible design space, while the environmental requirement indicates how a system should perform when it is exposed to a certain environment. Finally, the suitability requirements include all the '-ilities', like for example, maintainability, producibility, reliability, etc. [14]. Each element in the pattern is implemented as an object within the RVF. The class diagram showing the relations between the different objects is shown in Figure 4 (inspired on [13]). Using this approach, the requirements can be interpreted by software programs.

- **Performance**
  The SYSTEM shall FUNCTION with PERFORMANCE [and TIMING upon EVENT TRIGGER] while in CONDITION
- **Design (constraint)**
  The SYSTEM shall [exhibit] DESIGN CONSTRAINTS [in accordance with PERFORMANCE while in CONDITION]
- **Environmental**
  The SYSTEM shall [exhibit] CHARACTERISTIC during/after exposure to ENVIRONMENT [for EXPOSURE DURATION]
- **Suitability**
  The SYSTEM shall exhibit CHARACTERISTIC with PERFORMANCE while CONDITION [for CONDITION DURATION]

Fig. 3 Requirement patterns as shown in [13]. Patterns are originally from [14]. The parts between square brackets are optional

B. Requirement Verification Methods

Each requirement needs to have a verification method that can be used to check whether a design complies with the requirement. Note that a requirement needs to be quantifiable to be verified. If a requirement is not directly verifiable, lower-level requirements must be derived from the top-level requirement until a suitable verification method can be assigned. Within the RVF, a verification method consists of two elements: a means of compliance and a test case.

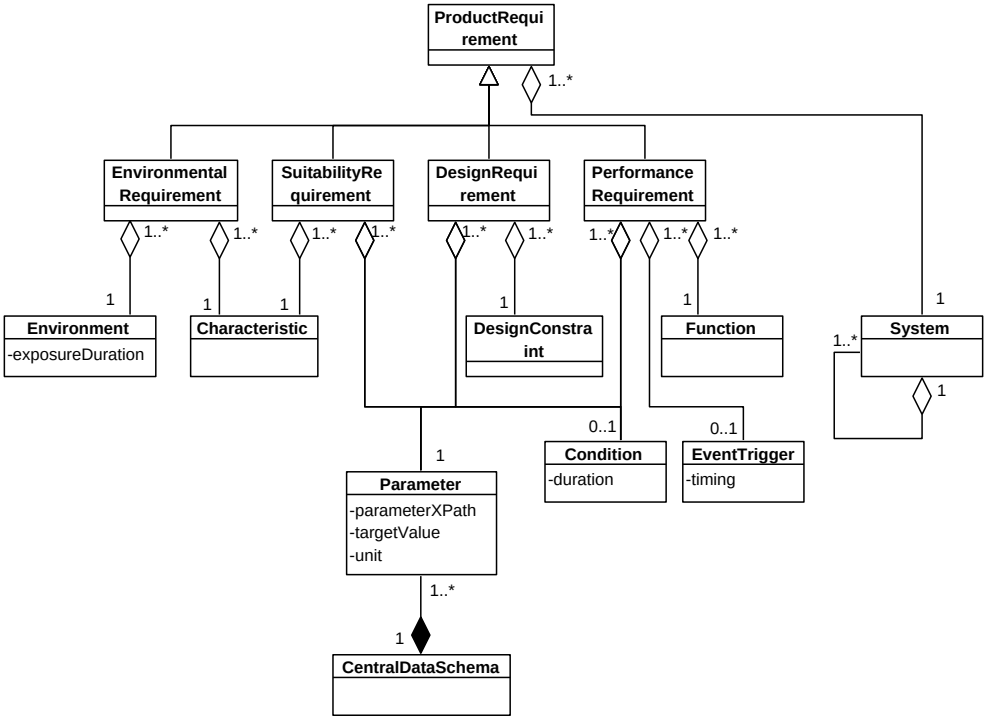

Fig. 4 Class diagram indicating how the requirements patterns are translated into classes within the RVF

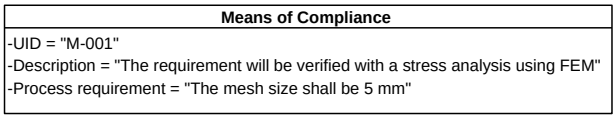

Fig. 5 Example of a Means of Compliance [15]

A means of compliance is defined as the agreement between the 'need stakeholder' and the 'responsible stakeholder' on how compliance will be achieved. The need stakeholder is the stakeholder that has the need from which the requirement is derived, while the responsible stakeholder is the stakeholder that is responsible for ensuring the design complies with the requirement. Within the means of compliance, agreements are made on for example software tools and versions, acceptable assumptions, physical tests, or test conditions. An example of a means of compliance is shown in Figure 5. Note that one requirement can have multiple means of compliance depending on for example the design stage. An example is shown in Figure 7. In this case, the rib cost may be verified using empirical cost estimation during the conceptual design stage, while during the preliminary design stage process-based cost estimations are required.

A test case is the technical implementation of the means of compliance. A test case consists of all the models, analysis tools, and physical tests (including the required input and output variables) that are necessary to verify the compliance of a requirement given a fixed design. At this time, the RVF focuses mostly on the conceptual and preliminary design stages. Therefore, only the virtual product models and analysis tools have been implemented. The product models (e.g. CAD model) and analysis models (e.g. FEM or CFD analysis) are called design competences. Thus, a test case within the RVF consists of design competences and their associated input and output variables. Figure 6 shows two examples of test cases. Indeed, as can be seen in the second test case, a test case can in itself be a small MDA workflow. In the future, the RVF could be extended to also support the setup of physical tests.

Figure 7 shows an example of two requirements and their associated means of compliance and test cases. As stated earlier, one requirement can have multiple means of compliance. Similarly, one means of compliance can have multiple test cases. For instance, in this example, the means of compliance agrees on using process-based cost estimations for the rib calculations in the preliminary design stage. Two cost models are available that comply with this agreement, so both can be used to estimate the rib costs. Note also that similar test cases can be used for different means of compliance. As shown in Figure 7, cost model 2 can be used for both the rib as well as the spar cost estimations. The difference between the two test cases is the required input and calculated output.

With the definition of a means of compliance, also verification process requirements are defined. These requirements focus on the verification process instead of the system of interest. They indicate the conditions to which the verification process must comply. An example is shown in Figure 5, where a requirement is placed on the mesh size of the FEM analysis tool. Verification process requirements determine the settings within a test case. For example, they can specify certain input values or the design competences to be used within a test case.

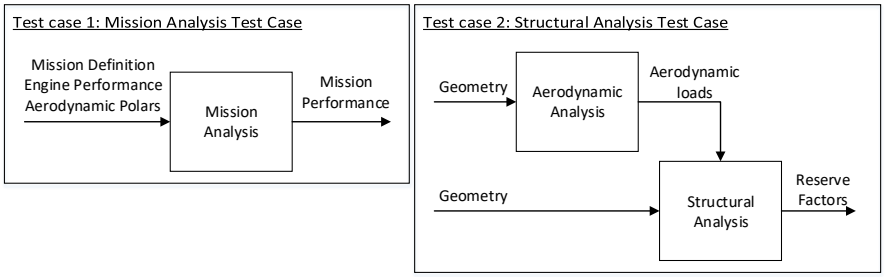

Fig. 6 Examples of test cases. A test case within the RVF consists of the design competences and associated input and output variables to verify the compliance of a requirement given a fixed design. A design competence is either a product model (e.g. CAD model) or an analysis tool (e.g CFD or FEM analysis)

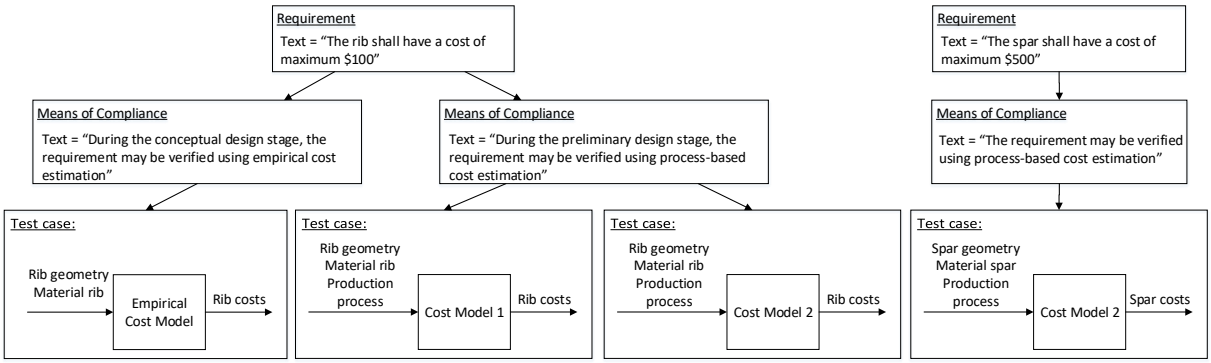

Fig. 7 Example indicating the link between requirements, means of compliances and test cases. One requirement can have multiple means of compliances attached to it based on for example the design stage. Furthermore, a means of compliance can have multiple test cases and test cases may be reused for the compliance verification of different requirements (with different input and output)

The class diagram in Figure 8 indicates the different elements and their relations concerning the (process) requirements, means of compliance, and test cases as implemented within the RVF. As can be seen in the Figure, a test case is built up from one or more design competences. These are the design competences that need to be executed to check the requirement's compliance. Furthermore, as indicated in Figure 8, the input and output of the design competences are specified by parameters. These parameters are the same parameters that are used within the requirement definitions (see Figure 4). This enables a direct connection between the requirement and its verification method and enables automatic verification of the requirement as the result from the test case can directly be compared to the value specified within the requirement.

C. Compliance Report

Once the means of compliance and test cases are determined for each requirement, a given design can be checked for requirement compliance. The first step is to select one means of compliance and test case per requirement. Next, the design can be loaded into the RVF. For each requirement, the corresponding test case is executed and the results are collected. The result values are then automatically compared with the target values that were specified within the

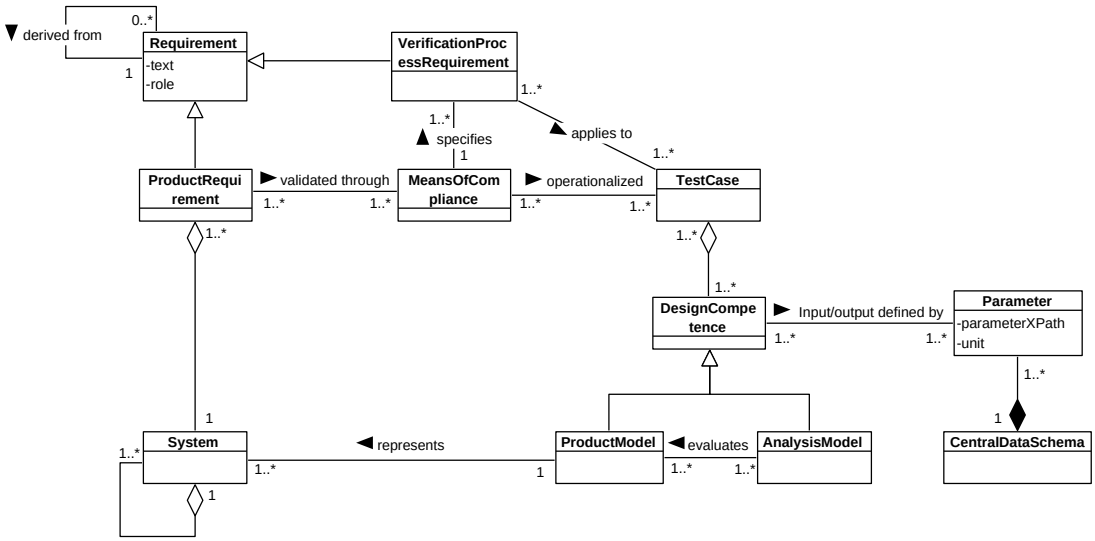

Fig. 8 Class diagram showing the relations between requirements, means of compliance and test cases. Furthermore, it indicates how a test case is built up from one or more design competences that are connected to a central data schema

requirements. By comparing the actual values with the target values, requirement compliance is determined. The results are then automatically reported in a compliance report.

Figure 9 shows an example of such a requirement compliance report, automatically generated using the RVF. The report shows the requirement, the design value, and the requirement compliance. Furthermore, the report indicates a compliance margin (indicated in Figure 9 as 'Difference'). The compliance margin indicates the percentual difference between the actual value and the target value of the requirement. A negative compliance margin indicates that the requirement has not been satisfied.

	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-1005	single bracket cost	The movable shall have a single bracket cost of less than $400	True	312.5	$	21.88%
R-1006	total skin cost	The movable shall have a total skin cost of less than $1500	True	1396.89	$	6.87%
R-1003	total mass	The movable shall have a total mass of less than 50 kg.	True	19.38	kg	61.24%
R-2005	root chord	The movable shall have a root chord of less than 3000 mm.	False	3537.21	mm	-17.91%
R-2006	tip chord	The movable shall have a tip chord of less than 3000 mm.	True	1507.25	mm	49.76%
R-2007	span	The movable shall have a span of less than 3000 mm.	False	3757.34	mm	-25.24%

Fig. 9 Example of an automatically generated requirements compliance report

IV. Requirements to Drive the Design Process

The previous section explained how requirement compliance can be checked for a given design. The goal of the second part of the framework is to actively use the requirements to guide the design process, formulated as an MDAO workflow. In this part, the design is iteratively changed to guarantee requirement compliance while simultaneously optimizing the design for a certain objective. As shown in Figure 1, the process consists of two steps. Section IV.A explains how different MDAO problem roles can be assigned to requirements to implement them in the MDAO workflow. Section IV.B describes how the MDAO workflow can be formulated and executed.

A. Assigning Roles to Requirements

Each requirement puts a restriction on the feasible design space. However, there are multiple ways on how this restriction can be taken into account in an MDAO workflow, as a requirement does not necessarily have to be implemented as a constraint. Within the RVF, six problem roles have been defined that can be assigned to each requirement and that determine how the requirement will be implemented in the MDAO workflow:

1) **Design variable**: A requirement can specify the allowed options for a design variable.
   Example: The flap shall be produced using in-house manufacturing methods.
   In this case, the manufacturing method can be implemented as a categorical design variable. Only those manufacturing methods that are available in the production facility can be used to manufacture the flap.
2) **Design variable bound**: A requirement can specify the upper and/or lower bound of a design variable.
   Example: The wing shall have a span less than xx m.
   If the wingspan is a design variable in the MDAO design process, then this requirement can be implemented as an upper bound on the wingspan variable.
3) **Input parameter**: A requirement can specify a fixed value for an input parameter.
   Example: The wing shall withstand all critical load cases.
   A requirement can be satisfied by setting the correct input or boundary conditions for an analysis tool. In this case, the load factor should be set to 2.5.
4) **Constraint**: A constraint limits the feasible design space.
   Example: The flap shall weigh less than xx kg.
   A requirement can be added as a constraint to limit the design space.
5) **Objective**: The objective indicates the quantity of interest that should be minimized or maximized during the optimization process.
   Example: The flap shall cost less than $xx
   The parameter of a requirement can be added as an objective to the MDAO problem.
6) **Quantity of Interest**: A quantity of interest does not put any limitation on the design space. It is a result one is interested in knowing.
   A typical design case can contain thousands of requirements. It may not be feasible to integrate all requirements in the optimization problem or the test case associated with a given requirement may be very expensive to evaluate. Therefore, a requirement can also be labeled as a quantity of interest. In this case, the associated test case will be executed after the optimization to check the requirement's compliance.

The problem role that is assigned to a requirement has a major impact on whether requirement compliance is guaranteed. If the requirement is implemented as a design variable (bound) or input parameter, then all design options the optimizer explores are compliant with the requirement by design. If the requirement is implemented as a constraint, the optimizer will evaluate design options for which the requirement can be violated. However, if the optimization is successful, the design will be compliant with the requirement. If the requirement is implemented as an objective or quantity of interest, there is no guarantee that the requirement will be satisfied by the final design.

Note that there can be multiple problem role options that can be assigned to a single requirement. It is up to the user to decide which problem role is assigned to each requirement. This means that different optimization and design problems can be derived from the same set of requirements. Not only optimization problems but also single design analyses or DoE's can be formulated. This provides the user with the flexibility to explore different what-if scenarios. If a requirement does not get a problem role assigned, it will also not be considered in the analysis.

B. Formulating and Executing the MDAO Workflow

Once the problem roles are assigned to the requirements, the next step is the formulation of the MDAO workflow. The class diagram in Figure 10 indicates the connection between the MDAO workflow and the test cases. For each requirement that has a problem role assigned to it, the test cases are collected, as the test cases contain the design competences that need to be implemented within the MDAO problem.

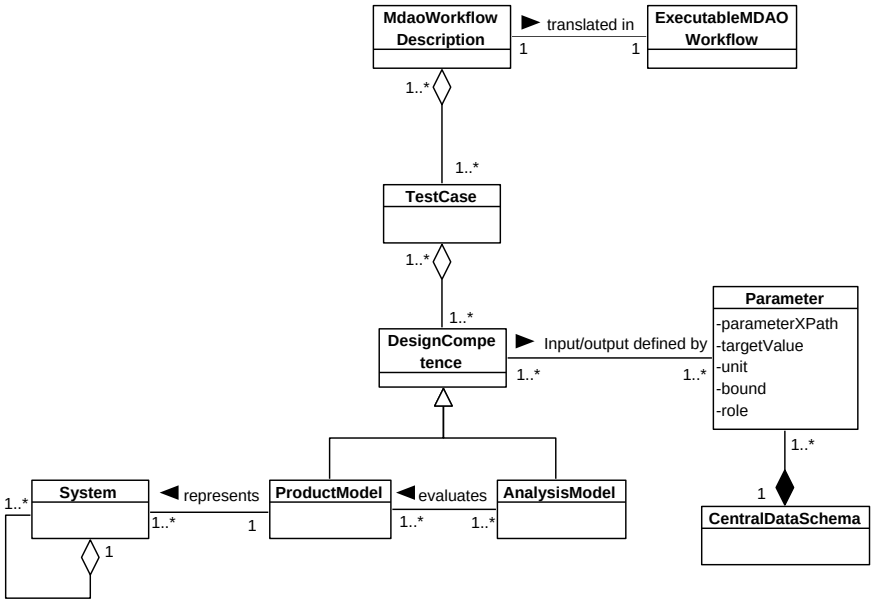

Fig. 10 Class diagram showing the relations between the MDAO workflow description and the test cases. Only the test cases from the requirements that have a problem role assigned to them will be taken into account in the MDAO workflow

The design competences from the different test cases can automatically be connected as the input and output of all competences are formulated according to a central data schema. This means that all competences require an input XML file and generate an output XML file according to the same data schema. This enables a plug-and-play method for the design competences as is shown in Figure 11. Besides the design competences, also the requirements use this central data schema, as shown in Figure 4. Each parameter in a requirement has an attribute called *parameterXPath*. This attribute points to an element in the central data schema. If a parameter is required that needs to be calculated from a combination of elements in the data schema, a wrapper needs to be created that maps the parameter to the elements of the data schema. Using a central data schema makes it easier to connect the requirements and design competences and to reuse them in different design studies.

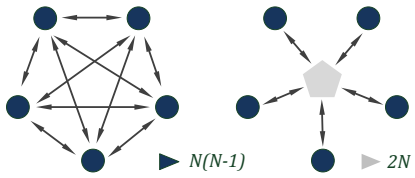

Fig. 11 Number of interfaces between disciplinary tools when using discipline specific interfaces (left) or when using a central data schema (right) (adapted from [16])

After the required design competences are identified, the MDAO workflow is formulated. Over the past few years, new methodologies have been developed that enable one to go from a repository of tools to fully formulated MDAO workflows [3–5]. These methodologies support the user in collecting the design competences, assigning problem roles (e.g. objective, constraint, design variable) to the input and output variables, and applying the MDAO architecture (e.g. DoE, MDF, IDF). If multiple requirements use the same design competences (for example with the Cost Model in Figure 7), they will also make sure that each design competence will appear only once in the MDAO workflow to prevent unnecessary calculations.

When the workflow has been formulated, it can be inspected by the user for correctness and completeness. Each element in the workflow is present because of a direct link with one of the requirements (through the problem roles and test cases). If some competences the user was expecting are missing, this could be an indication that some requirements

are missing or that certain test cases are incomplete. For example, if the user expects a structural analysis tool within the workflow, but the tool is missing, it could be that a requirement concerning the structural properties of the design is missing or that the structural analysis tool is missing in a test case. Besides missing competences, it could also be that some extra design competences are required to calculate some of the required input values. Finally, the user can add extra design variables or constraints if desired.

Once the MDAO workflow has been inspected by the user, it needs to be translated into an executable workflow such that results can be obtained. A methodology to automate this process has been developed by Van Gent and La Rocca [4]. In this case, the formulated workflow is written to a CMDOWS (Common MDO Workflow Schema) file [17], which can then be imported into a PIDO (Process Integration and Design Optimization) tool, like RCE[6] or OpenMDAO [18], to generate the executable workflow. Once the MDAO workflow has been executed, the results can be imported back into the RVF to generate a requirements compliance report, as described in Section III.C.

V. Prototype Implementation

Figure 12 shows an overview of all elements in the RVF and their relations. The class diagram shows the derivation of the requirements from the stakeholders, the formulation of the machine-readable requirements, the connection of the requirements with their verification methods, and the derivation of the MDAO workflows from the test cases. The different parts are indicated with the colored boxes. Based on this overview, a first prototype of the RVF has been implemented into a software package as described in the work of Van Manen [15].

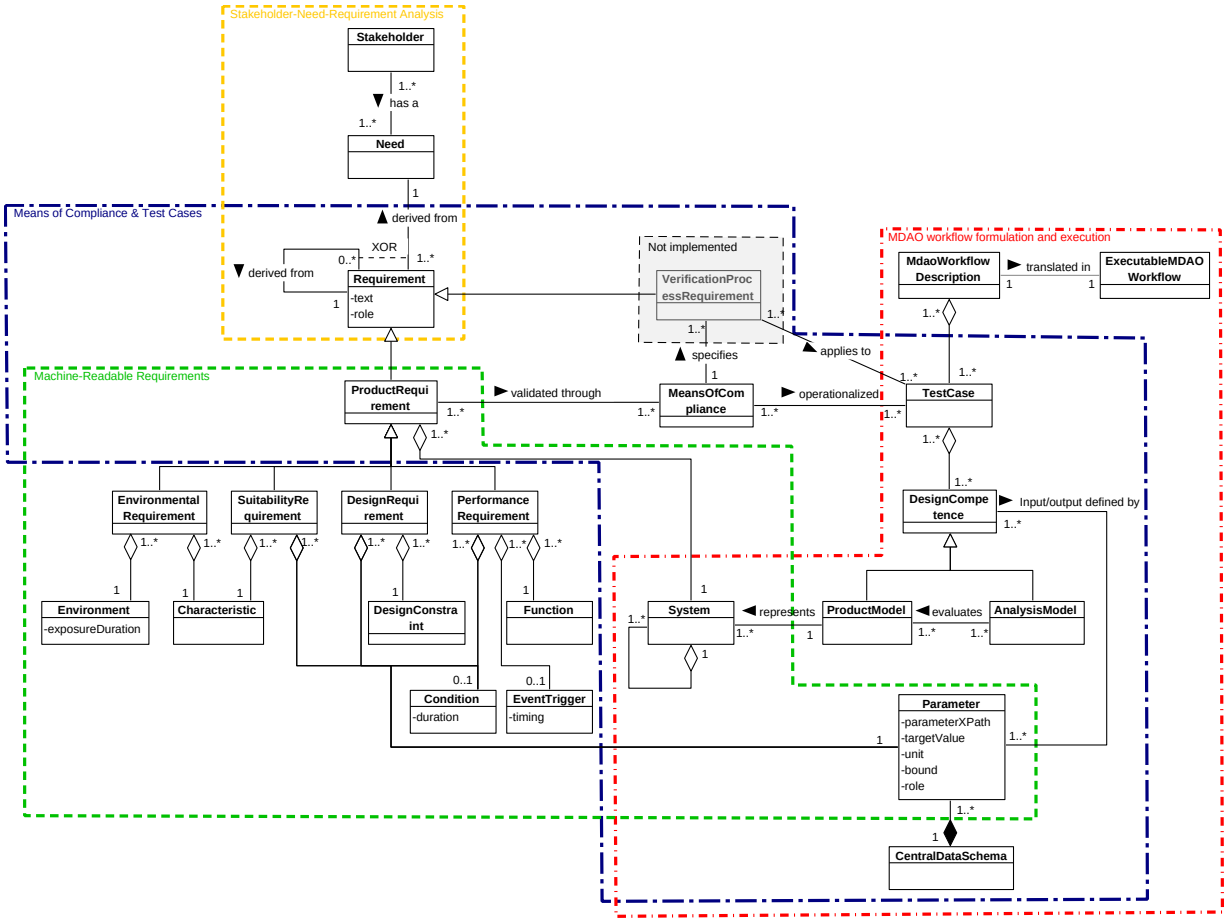

Fig. 12 Class diagram showing all the elements of the RVF and their relations

[6] `https://rcenvironment.de/`, accessed on: 29-04-2022

The prototype has been implemented in a Knowledge-Based Engineering (KBE) application built using the ParaPy[7] system. The use of KBE technology provides several advantages. Due to ParaPy's GUI, one can easily inspect the relations between the requirements, means of compliance, test cases, and requirement compliance. Furthermore, one can directly execute the test cases from within the KBE application. Due to the dependency tracking, only those design competences are executed that are necessary to calculate the compliance for the requested requirements. Because of the result caching and lazy evaluation, the design competences are only executed once even though they may be required to check the compliance of several requirements, which saves time and computational resources. As soon as some requirement changes, the previously generated data dependent on the changed requirement is invalidated and reevaluated on demand.

Note that not all elements as shown in Figure 12 have been implemented in the prototype yet. For example, the verification process requirements that put restrictions on the test cases, are not yet supported.

VI. Results

The prototype of the RVF was applied to a use case of a Tier-1 supplier [15]. The use case focuses on the design of a wing box for an aircraft fin. An example of such a wing box is shown in Figure 13. The wing box consists of two spars, multiple ribs, and a top and bottom skin.

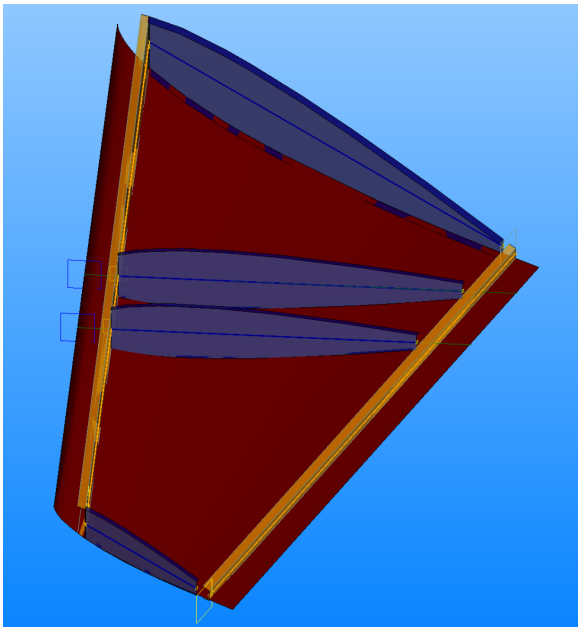

Fig. 13 Example of a wing box (generated using ParaPy) consisting of two spars (orange), multiple ribs (purple) and a top and bottom skin (red). The top skin is left out for clarity [15]

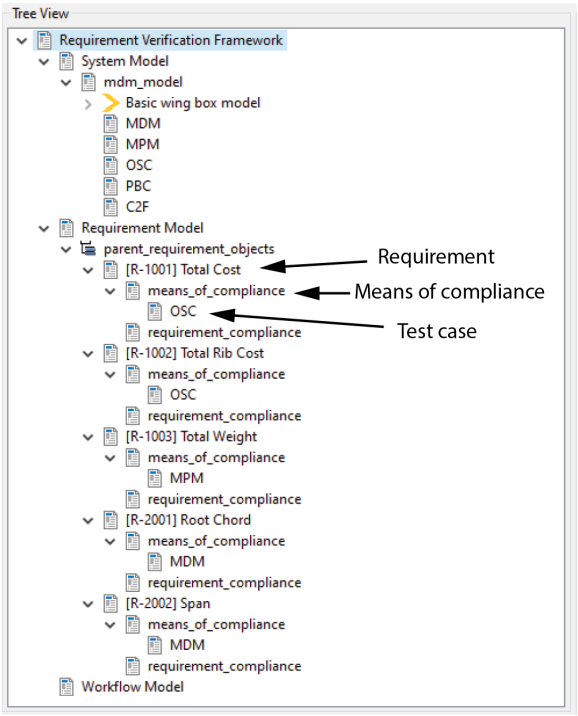

Fig. 14 Screenshot showing the requirements, means of compliance and test cases as implemented in the RVF [15]

The first step is to formulate the machine-readable requirements. Five requirements were formulated that put restrictions on the cost, weight, and geometry of the wing box. These requirements are listed in Table 1. The colors used within the requirement text correspond to the colors of the requirement patterns, as shown in Figure 3, to make the requirements machine-readable.

For each requirement, one or two means of compliance and test cases have been identified as shown in Figure 15. As explained in the previous section, the verification process requirements have not been implemented in the RVF yet, and are therefore also not included in the means of compliance. Four different disciplinary tools are used within the test cases. The Multidisciplinary Modeller (MDM) [19] is a KBE model and generates the CAD model of the wing box.

[7] `https://parapy.nl/`, accessed on: 21-04-2022

Table 1 Overview of the wingbox requirements. The colors indicate the different elements of the requirement patterns as specified in Figure 3 [15]

ID	Requirement text	Requirement type
R-1001	The wing box shall have a total cost of less than $7,000.	Performance
R-1002	The wing box shall have a total weight of less than 50 kg.	Performance
R-1003	The wing box shall have a total rib cost of less than $1,600.	Performance
R-2001	The wing box shall have a span in the range of 3.5 m and 4.0 m.	Design constraint
R-2002	The wing box shall have a root chord in the range of 3.0 and 3.5 m.	Design constraint

The Mass Properties Module (MPM) calculates the mass of the wing box. The Open Source Cost tool (OSC) [20] is an open-source tool that calculates the cost of the wing box, given the manufacturing method, geometry, and geometry complexity. The Project-Based Cost tool (PBC) is an alternative to the OSC tool.

The requirements, means of compliance, and test cases are implemented within the RVF. The resulting KBE tree can be seen in Figure 14. As shown in Figure 15, the requirement on the total cost of the wing box has two means of compliance and two test cases. Within the RVF, one can easily select one or the other depending on the scenario the user wants to evaluate. For now, the means of compliance during the conceptual design stage and the open-source cost estimation test case are chosen and selected within the RVF.

Once the means of compliance and test cases have been selected, a first compliance report can be generated based on a fixed design. The design parameters and corresponding values for the initial design are shown in Table 2. Based on these design parameters, the test cases are executed by the RVF and the compliance report is automatically generated. The results are shown in Table 3. As can be seen, three of the five requirements are violated. The initial wing box design is too heavy, the rib costs are too high and the root chord is too large.

Table 2 Design parameters of the initial wing box [15]

Parameter	Value	Unit
Root chord	3.54	m
Tip chord	1.51	m
Sweep	40.66	deg
Span	3.76	m

Table 3 Automatically generated compliance report for the initial design of the wing box. The report indicates the calculated values, indicates whether the requirement has been met and shows the compliance margin. [15]

ID	Requirement text	Test case value	Compliant	Compliance Margin
R-1001	The wing box shall have a total cost of less than $7,000.	$6,824	True	2.51%
R-1002	The wing box shall have a total weight of less than 50 kg.	57.97 kg	False	-15.94%
R-1003	The wing box shall have a total rib cost of less than $1,600	$1,710	False	-6.88%
R-2001	The wing box shall have a span in the range of 3.5 m and 4.0 m.	3.76 m	True	7.43 / 6.00%
R-2002	The wing box shall have a root chord in the range of 3.0 and 3.5 m.	3.54 m	False	18.00 / -1.14%

One of the benefits of the RVF is that a design process can be formulated to change the current wing box design to try and find a design that complies with the specified requirements, while simultaneously optimizing the design for a specified objective. The MDAO problem roles that are assigned to the different requirements are shown in the third column of Table 5. In this case, the wingspan and root chord have been chosen as design variables and therefore their corresponding requirements (R-2001 and R-2002, respectively) are marked as design variable bounds. Note, that the tip chord and sweep have not been chosen as design variables and will stay constant during the optimization. The requirement on the total cost of the wing box (R-1001) is marked as a constraint and the objective is to minimize the weight of the wing box (R-1002). The rib cost requirement (R-1003) will not be enforced during the design process as it is marked as a quantity of interest. The resulting MDAO workflow, automatically generated using KADMOS [4], is shown in Figure 16. KADMOS (Knowledge- and graph-based Agile Design for Multidisciplinary Optimization System) is one of the software tools that can automatically generate an MDAO workflow starting from a repository of tools.

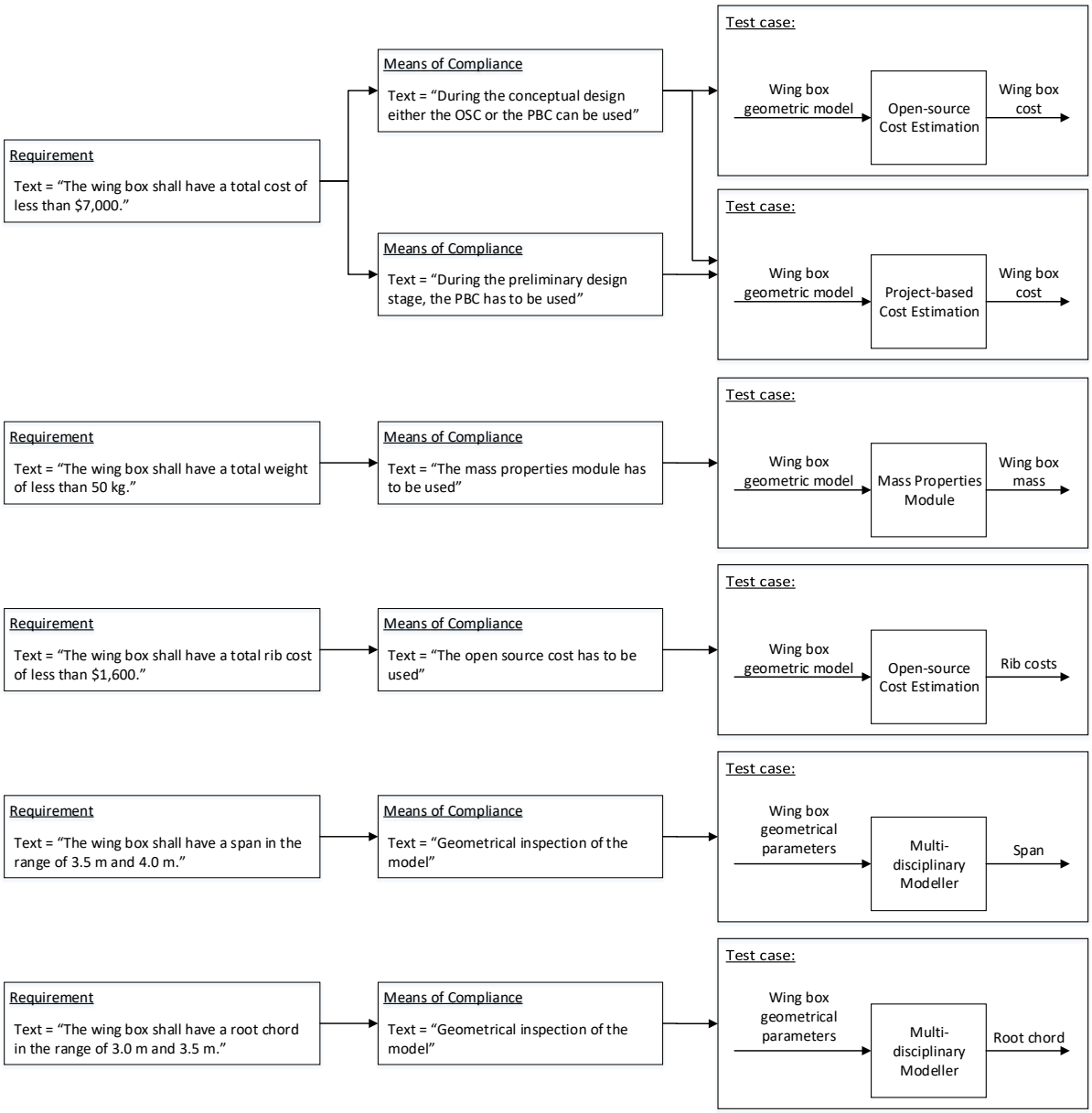

Fig. 15 Overview of the five requirements applied to the wingbox, including the connected means of compliances and test cases

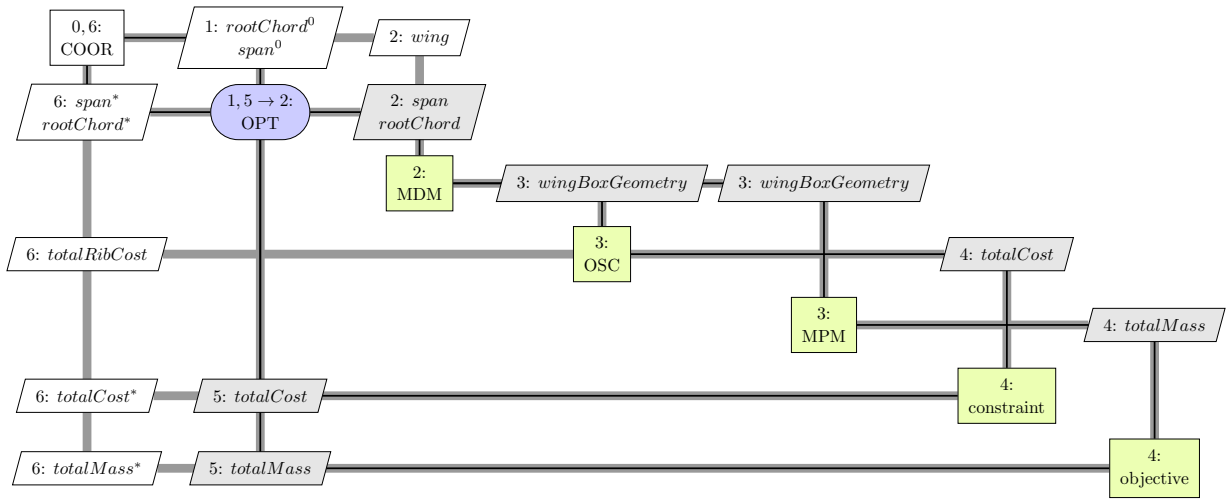

Fig. 16 XDSM generated using KADMOS. The optimization focuses on minimizing the mass of the wing box [15]

Once the MDAO workflow has been formulated, the user can inspect the workflow. In this case, one can see that important elements are missing. For example, there is no disciplinary tool concerning the structural analysis of the wing box present in the MDAO workflow. None of the requirements needed this tool, therefore the tool was not presented in the test cases used to built up the MDAO workflow. This is an indication to the user that important requirements are missing, for example: "*The wing box shall withstand all critical load cases.*"

The MDAO workflow as defined by KADMOS was translated into an executable workflow by uploading the CMDOWS file into RCE. The optimization was executed and the results were loaded back into the RVF. The values of the optimized wing box are shown in Table 4. Within the RVF, a requirement compliance report was again automatically generated and the results are shown in Table 5. Here, one can see that in this case the design is compliant with all requirements. The final design is smaller than the initial design and has a reduced cost and weight.

Table 4 Design parameters of the optimized wing box [15]

Parameter	Value	Unit
Root chord	3.00	m
Tip chord	1.51	m
Sweep	40.66	deg
Span	3.5	m

Table 5 Automatically generated compliance report for the final design of the wing box. The table also indicates the MDAO problem roles that were used to formulate the design process. [15]

ID	Requirement text	MDAO problem role	Test case value	Compliant	Compliance Margin
R-1001	The wing box shall have a total cost of less than $7,000.	Constraint	$6,017	True	14.04%
R-1002	The wing box shall have a total weight of less than 50 kg.	Objective	48,28 kg	True	3.44%
R-1003	The wing box shall have a total rib cost of less than $1,600	Quantity of Interest	$1,538	True	3.88%
R-2001	The wing box shall have a span in the range of 3.5 m and 4.0 m.	Design variable bound	3.50 m	True	0.00 / 12.50%
R-2002	The wing box shall have a root chord in the range of 3.0 and 3.5 m.	Design variable bound	3.00 m	True	0.00 / 14.29%

VII. Conclusions and Outlook

In aircraft design, requirements must be well managed to ensure the final product meets the needs of all the stakeholders. Nowadays, requirements are often managed using dedicated software packages. However, these software packages are limited to a direct evaluation of the requirements given a fixed design. What-if scenarios can only be evaluated by manually changing the design and analyzing the effect on requirements, which is time-consuming and leads to sub-optimal designs.

This paper presented a new framework, called the Requirements Verification Framework, that focuses on the implementation of model-based requirements in the aircraft design process using MBSE and MDAO. The framework enables the automatic verification of requirements and lets requirements drive the design process. By iteratively and automatically changing the design, requirement compliance is enforced while simultaneously optimizing the design for a certain objective.

The RVF consists of a five-step approach. During the first step, machine-readable requirements are formulated. Next, verification methods are defined for each requirement. Verification methods contain the methods required to check the compliance of a requirement. In the third step, MDAO problem roles, such as objective, design variable, constraint, or quantity of interest are assigned to the requirement. Based on the MDAO problem roles and the verification methods, the full MDAO workflow can be formulated and executed in the fourth step. Finally, once the MDAO workflow has been executed, the results are automatically checked for compliance and a compliance report is generated.

The framework was implemented in a software prototype, which was applied to the design of a wing box. The use case demonstrated how requirements are formulated, how they are implemented in the MDAO process, and how they are verified. By deriving an optimization problem from the requirements, an improved wing box design was obtained that satisfied all specified requirements, while simultaneously minimizing the weight of the wing box. Furthermore, the use case demonstrated how the RVF can be used to inspect the correctness and completeness of both the requirements as well as the design problem. The software prototype is not complete yet. Verification process requirements will be implemented in the future.

The RVF is completely model-based and object-oriented. This improves the traceability from requirements to product design, as all stakeholders can inspect the models to see how the design process was formulated and how requirement compliance has been achieved. Furthermore, it enables the reuse of models within different design studies, reducing the amount of rework required.

With the framework, a direct link between MBSE and MDAO has been established. This enables the easier evaluation of what-if scenarios leading to better design choices. Most importantly, using the framework, optimized designs can be obtained that comply with all the stakeholders' needs.

In the future, the RVF will be further extended to also include design-dependent requirements. As soon as a design choice has been made to include a certain subsystem in the design (e.g. a turbofan engine), new derived requirements become active. These derived requirements would have stayed inactive if a different design choice was made (e.g. a propeller instead of a turbofan engine). Therefore, the RVF will be extended to include the possibility to automatically activate requirements based on design choices. This will make it easier to compare design alternatives, leading to better design choices.

Besides the design-dependent requirements, also manufacturing requirements will be included in the RVF. The production process puts requirements on the design and vice versa, creating a system of systems. By taking the interactions between design and production into account already during early design stages, improved designs can be obtained that are also producible.

Acknowledgments

The research presented in this paper has been performed in the framework of the AGILE 4.0 project (Towards Cyber-physical Collaborative Aircraft Development) and has received funding from the European Union Horizon 2020 Program under grant agreement n° 815122.

References

[1] Smith, A., "PMI's Pulse of the Profession: Requirements Management - A Core Competency for Project and Program Success," Tech. rep., Project Management Institute, 2014.

[2] 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.

[3] Hoogreef, M. F. M., "Advise, Formalize and Integrate MDO Architectures: A Methodology and Implementation," Ph.D. thesis, Delft University of Technology, 2017.

[4] Van Gent, I., and La Rocca, G., "Formulation and integration of MDAO systems for collaborative design: A graph-based methodological approach," *Aerospace Science and Technology*, Vol. 90, 2019, pp. 410–433.

[5] Page Risueño, A., Bussemaker, J. H., Ciampa, P. D., and Nagel, B., "MDAx: Agile Generation of Collaborative MDAO Workflows for Complex Systems," *AIAA Aviation Forum*, 2020.

[6] Cencetti, M., Pasquinelli, M., and Maggiore, P., "System Modeling Framework and MDO Tool Integration: MBSE Methodologies applied to Design and Analysis of Space System," *AIAA Modeling and Simulation Technologies (MST) Conference*, 2013.

[7] Jeyaraj, A., Tabesh, N., and Liscouët-Hanke, S., "Connecting Model-based Systems Engineering and Multidisciplinary Design Analysis and Optimization for Aircraft Systems Architecting - A case study within the AGILE4.0 project," *AIAA Aviation Forum*, 2021.

[8] Leserf, P., de Saqui-Sannes, P., Hugues, J., and Chaaban, K., "SysML Modeling for Embedded Systems Design Optimization," *3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, 2015.

[9] Beernaert, T. F., and Etman, L. F. P., "Multi-level Decomposed Systems Design: Converting a Requirement Specification into an Optimization Problem," *Proceedings of the Design Society: International Conference on Engineering Design*, Vol. 1, No. 1, 2019, pp. 3691–3700.

[10] International Organization for Standardization, "ISO/IEC/IEEE 24765 - Systems and software engineering - Vocabulary," , 2017.

[11] U.S. Department of Transportation, "Systems Engineering Guidebook for Intelligent Transportation Systems - Version 3.0," , 2009.

[12] Bussemaker, J. H., Ciampa, P. D., and Nagel, B., "System Architecture Design Space Exploration: An Approach to Modeling and Optimization," *AIAA Aviation Forum*, 2020.

[13] Boggero, L., Ciampa, P. D., and Nagel, B., "An MBSE Architectural Framework for the Agile Definition of System Stakeholders, Needs and Requirements," *AIAA Aviation Forum*, 2021.

[14] Carson, R. S., "Implementing structured requirements to improve requirements quality," *INCOSE International Symposium*, Vol. 25, No. 1, 2015, pp. 54–67.

[15] Van Manen, H. S., "Development and Implementation of an MBSE Requirement Verification Framework in the System Design Process," Master thesis, Delft University of Technology, 2022.

[16] Alder, M., Moerland, E., Jepsen, J., and Nagel, B., "Recent Advances in Establishing a Common Language for Aircraft Design with CPACS," *CEAS - Aerospace Europe Conference 2020*, 2020.

[17] Van Gent, I., La Rocca, G., and Hoogreef, M. F. M., "CMDOWS: a proposed new standard to store and exchange MDO systems," *CEAS Aeronautical Journal*, Vol. 9, No. 4, 2018, pp. 607–627.

[18] Gray, J. S., Hwang, J. T., Martins, J. R. R. A., Moore, K. T., and Naylor, B. A., "OpenMDAO: An Open-Source Framework for Multidisciplinary Design, Analysis, and Optimization," *Structural and Multidisciplinary Optimization*, Vol. 59, 2019, pp. 1075–1104.

[19] van den Berg, T., and van der Laan, T., "A multidisciplinary modeling system for structural design applied to aircraft moveables," *AIAA Aviation Forum*, 2021.

[20] van der Laan, T., Johman, A., van Puffelen, T., Nolet, S., van Maanen, B., Daugulis, E., and van den Berg, T., "An open source part cost estimation tool for MDO purposes," *AIAA Aviation Forum*, 2021.