



FUSE: A Novel Design Space Exploration Method for Aero Engine Components That Combines Functional and Physical Domains

Downloaded from: <https://research.chalmers.se>, 2025-02-22 19:44 UTC



Citation for the original published paper (version of record):

Pradas Gómez, A., Panarotto, M., Isaksson, O. (2025). FUSE: A Novel Design Space Exploration Method for Aero Engine Components That Combines Functional and Physical Domains. *Aerospace*, 12(1). <http://dx.doi.org/10.3390/aerospace12010051>

N.B. When citing this work, cite the original published paper.

Article

FUSE: A Novel Design Space Exploration Method for Aero Engine Components That Combines Functional and Physical Domains

Alejandro Pradas Gómez ^{*}, Massimo Panarotto  and Ola Isaksson 

Product Development, Department of Industrial and Materials Science, Chalmers University of Technology, 412 96 Gothenburg, Sweden; massimo.panarotto@chalmers.se (M.P.); ola.isaksson@chalmers.se (O.I.)

* Correspondence: alejandro.pradas@chalmers.se

Abstract: Society awareness and environmental goals are forcing the aerospace industry to develop new sustainable system architectures. The components in the new system have to meet new functional requirements using alternative technologies and design solutions while ensuring that the physical performance of the component is maintained. However, design space exploration of both domains is challenging due to the intrinsic differences and nature of each: functional domain exploration deals with alternative means to solve functions, while physical exploration deals with parametric values, such as geometric dimensions and material types. Here, we present a method that enables concurrent exploration of the functional and physical design space. The method is based on a review of existing design space exploration methodologies. It has been developed in collaboration with industry and validated within a use case. We expect that this method will be useful for designers in conceptual phases where there are several functions containing multiple design alternatives and incompatibilities among them. The results of the method will allow designers to narrow down the design space to a few architectural candidates, including a baseline of physical dimensioning for each candidate.

Keywords: product design; design space exploration; product architecture; enhanced function-means (EF-M); knowledge-based engineering (KBE); design automation



Academic Editor: Yaolong Liu

Received: 30 September 2024

Revised: 3 December 2024

Accepted: 6 January 2025

Published: 13 January 2025

Citation: Pradas Gómez, A.; Panarotto, M.; Isaksson, O. FUSE: A Novel Design Space Exploration Method for Aero Engine Components That Combines Functional and Physical Domains. *Aerospace* **2025**, *12*, 51. <https://doi.org/10.3390/aerospace12010051>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the development of new aircraft programs, new propulsion architectures and technologies are required to meet challenging environmental goals (e.g., ACARE 2050 [1] and WayPoint 2050 [2]). These architectures must deliver increased efficiency while maintaining airworthiness levels and must be developed in a short timescale to have an impact by 2050 [3].

The traditional commercial aerospace industry is composed of different players: original equipment manufacturers (OEMs) and risk sharing partners (RSPs). These are two complementary viewpoints that need to be treated concurrently. OEMs, such as Airbus, Boeing, Rolls Royce, or General Electric, are in charge of the overall system architecture and integration. In contrast, RSPs such as GKN Aerospace focus on the design of the physical component design, their performance, and their manufacturing process. This artificial binary segmentation of roles is useful in guiding the readers into two mindsets. In reality, the authors acknowledge that OEMs develop components in-house, such as the jet core engine, and RSP also performs system architecture decisions when the delegation of authority is not only for a component but also for an entire subsystem. These two

perspectives, system engineering and component design, and their different approaches to exploring the design space [4], are the focus of this paper. Visualization of both domains is conceptualized in Figure 1.

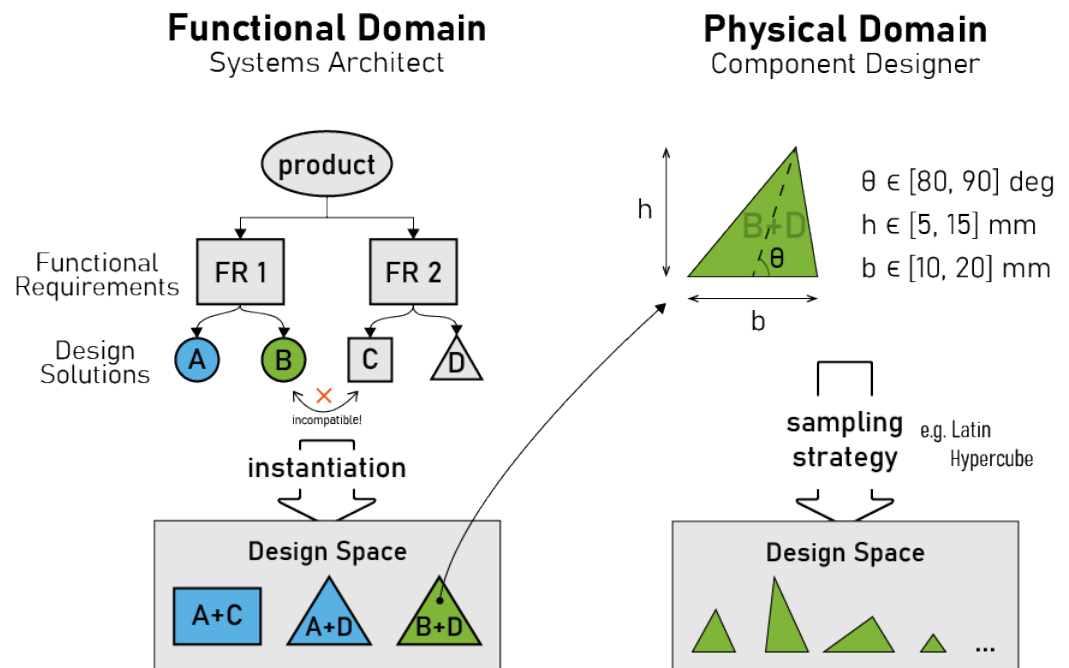


Figure 1. Visualization of the different nature of the design space exploration. On the left, the functional domain is explored by system architects. On the right, the physical domain is explored by component designers.

To be able to develop an innovative component that fits the novel architectures, RSPs now need to expand their component boundaries and understand the architectural trends of the overall system architecture (engine or aircraft) [5]. The novelty in future system architectures will be delivered by a mix of traditional components that perform new functions, or mature technologies in other sectors will be applied in the aerospace sector. For example, technologies can influence the function of the product (use hydrogen instead of fossil fuels [6]), the manufacture of the component (use additive manufacturing instead of traditional methods [7]), or both (additive manufacturing allowing the manufacture of specific combustion chambers for hydrogen [8]).

Design space exploration (DSE) is the exploration of design alternatives with the support of a computer [9]. This design activity, also called *tradespace* exploration [10], includes the systematic evaluation of alternatives according to an appropriate metric. DSE is defined here as a systematic evaluation of design concepts that supports the identification of one or more promising design candidates within a bounded domain. The importance of simultaneously exploring the entire system-level solution (architectures) while gaining sufficient knowledge at the component design level is emphasized.

Ulrich defined architectures as the scheme by which the function of the product is mapped onto physical components [11]. In particular, the product architecture is defined by (1) the arrangement of *functional elements*, (2) the mapping from *functional elements* to *physical components*, and (3) the specification of the *interfaces* between interacting physical components. Therefore, when talking about system architecture, designers talk about the functional domain.

The perspective from which designers approach the development of products is subjective and recursive. It is subjective because what is considered a system or a component (subsystem) depends on the designer's responsibility. For example, aircraft designers

(e.g., Airbus) consider their product, the system, and the engine as one of the components. From the engine designers' perspective (e.g., Rolls-Royce), the engine is a system, and its modules (fan, compressor, combustion chamber, etc.) are considered components. Looking up to the recursion tree, an aircraft may be considered a component in a system-of-systems approach, along with the other systems for efficient airport and traffic control operations. It is recursive because the system and component approach can be applied at any of the levels described above. Most importantly, once a perspective is set, the development methodology is as follows: system engineering [12] or component design [13]. The example in this research takes the systems perspective on an engine and the component design perspective on an engine part. The recursive property could potentially be applied again, and it is considered that the engine component is a system with subsystems, but this is not the case here.

The relation between the architectural domain (functional) and the component (physical) domains is highly iterative in a novel product design. For simplicity, the product development process models have decoupled the space exploration in each domain and presented them as sequential activities: "Concept" vs. "Detailed Design" by Ulrich and Eppinger [13] or "Architecting Definition Process" vs. "Design Definition process", as suggested by INCOSE [12]. However, for novel architectures, iterative mapping or "zig-zagging" between the function and the physical domains [14] plays a critical role, as expert judgment and previous design data may be misleading. System architects need detailed information to select concepts, and component designers need an overview of all possible architectures against which to evaluate their components. The two domains are coupled, as has been discussed in the engineering design community [15]. In addition, jet engine components are considered *functionally integrated* and rarely comply with traditional, physical, modular architectures [16], increasing their inter-dependencies and complexity.

Aircraft and engine system architects work within the functional domain [17]. Their responsibility in the design process is to translate the needs from the customer domain into the functional domain as functional requirements (FR) and to ensure that those requirements will be met by valid design solutions (DS) [14]. They work with the combinatorial alternatives of design solutions or categorical variables in their design space exploration: *Should I use design solution A or B to fulfill this requirement? Is technology A incompatible with technology C?* [18]. The evaluation of the technologies and their impact is based on previous aircraft or propulsion systems, or "top-level" models that capture the trends based on the dimensioning system parameters. The global effect of new technologies on architecture performance is accounted for with impact factors [19].

On the other hand, component designers work in the physical domain, which is what is being manufactured. They design a subsystem of a system, for example, a structural component within a jet engine. They need a baseline architecture to modify and evaluate its performance, so the design exploration is limited to a few architectures. This exploration can contain categorical variables (material and manufacturing processes) but is mainly focused on numerical variables, such as thicknesses, angles, or number of items. Their design space exploration strategies can be purely exploratory (like a Latin hypercube sampling strategy) or driven by an optimizer. The evaluation of the different designs is based on detailed multi-disciplinary performance metrics that use physical or geometrical models: CAD, CFD, or FEM, to name a few. Their final metrics are usually weight and cost and supporting values such as structural strength and aero-performance as constraints.

Obtaining accurate estimations of all the possible alternative solutions that are emerging is a challenge [18], but it is necessary to evaluate novel architectures. They require the generation, morphing, or combination of different physical models. The setup and automation of those workflows require expert knowledge on programming or the specific

automation tool, and often several manual debugging iterations and example inspections are required to get the automation running. The speed with which evaluations are to be performed needs to increase to meet the deadlines imposed above, for which automation is required. There are many automation strategies and techniques to generate many architectures quickly and to evaluate the performance of parametric changes for a given architecture.

The problem is that the system architecture behavior is dependent on the component behavior and vice versa, yet design approaches differ. This paper explores the techniques used to explore and evaluate different architectures using standard software tools. The baseline method used for comparison is Muller et al. [20], where a method is developed to generate physical models from the functional domain. The focus of Muller's research was to explore and connect the two domains. The focus of this research is design space exploration, for which performance values need to be linked to design solutions in order to evaluate and down-select architectures.

The research question is as follows: *How can the design exploration (including evaluation) of functional and physical domains be combined and automated?* The hypothesis is that a simple method to combine the exploration of functional and physical design space can increase the information exchange between both domains. The result of the research is a method that supports designers in physically evaluating different solutions in an exploding architecture configuration scenario. The novelty of the design exploration method, compared to the related work, is that it combines the physical evaluation of design alternatives with the generation of those configurations via the functional domain.

The purpose of this research is to develop and present a method for the concurrent exploration of the functional and physical domains. To demonstrate the method, a use case has been designed with the intention of simulating its application in a realistic scenario. It is important to note that this study does not aim to develop a final product; therefore, validation of specific analysis methods, such as detailed simulation results, is beyond the scope of this work.

Section 2 explores previous work done on bridging both domains. Section 3 presents the novel method, and Section 4 applies the method in an industrial case. Section 5 discusses the method and compares it to the existing methods, and Section 6 concludes the paper.

2. Review of the Related Work

This review section focuses on two key perspectives in the design process. The first perspective distinguishes between the functional and physical domains. The functional domain represents the space where designers define the functional requirements a product must fulfill to create value for users or stakeholders. In contrast, the physical domain involves the tangible aspects and physical variables of the design, such as geometric and structural details, often modeled using CAD tools.

The second perspective examines generation versus evaluation activities. In the product development process, designers first generate a design concept (e.g., an architecture or a physical product) and then evaluate the proposed concept against specific criteria.

By combining these two perspectives, we can identify four distinct design activities.

- **The generation of architectures** involves defining potential design structures, as discussed by Schachinger and others [21].
- **The evaluation of architectures** can occur without a physical model, for example, using methods like change propagation analysis to evaluate aircraft architectures or jet engine components [22].
- **The generation of physical models** for design space exploration can leverage, for example, knowledge-based engineering methodologies [23].

- **The evaluation of physical models** can use traditional modeling techniques such as FEM or CFD.

In this paper, we focus on the generation of architectures that necessitate some form of detailed physical evaluation. As noted in the introduction, this dual exploration is essential for addressing the complexity of design challenges. Figure 2 visualizes prior work and maps it to the two perspectives mentioned earlier, highlighting how different methods align with these four combinations and how they try to bridge the domains.

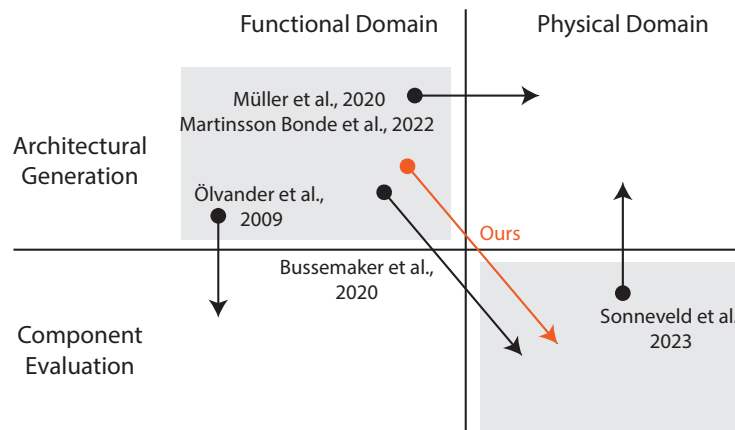


Figure 2. Visualization of recent approaches that bridge architectural vs. component and functional vs. physical domains. For clarity purposes, only representative papers from Section 2 that make a significant effort towards bridging the domains have been selected for visualization. These are: Müller et al. [24], Martinsson Bonde et al. [25], Ölvander et al. [26], Bussemaker et al. [27], Sonneveld et al. [28].

The body of knowledge on the connection of functional to physical domains is vast. The research has been conducted within an aerospace context and, as such, has a bias in its automation techniques and engineering design methodology. Three related areas have been researched. The first one is product development and functional perspective, which are relevant to the engineering design context and its design exploration techniques. The second area of research is model-based systems engineering (MBSE). This area is relevant because of its industrial adoption for the architectural definition in the functional domain. Finally, the third area of knowledge-based engineering (KBE) has been explored. KBE is relevant due to its ability to physically represent models and evaluate their performance. Moreover, its object-oriented architecture, coupled with the reusability of its features, positions KBE as a promising candidate for integration and automation within the functional domain.

2.1. Design Exploration from a Functional Product Perspective

Ölvander et al. [26] propose the use of morphological matrices to account for different variants. The aim of the design method is to provide aircraft system configurations. It considers the possible design solutions for each sub-function as an off-the-shelf component. Therefore, there is no consideration of optimizing each variant needed for a structural architecture. Computer-aided design (CAD) models are not generated in this process. The evaluation of design alternatives is automated and quantitative. However, the evaluation relies on the use of analytical formulas based on the different variant parameters. There is no consideration for the geometrically derived performance metrics, such as weight or stresses, that are required for the detailed evaluation of the component performance.

Instead of a morphological matrix, Müller et al. [29] proposed using enhanced function means (EF-M) [21] to conduct design space exploration. Each functional requirement (FR) may be fulfilled by only one design solution (DS). A tree with more than one DS under an FR represents a variant. The combination of all possible DS that fulfill the FR constitutes

the architectural design space. Later on, Müller et al. [24], and later in [20] proposed an approach to extend EF-M modeling to automate the generation of CAD representation of variants by linking the EF-M DS with the Siemens NX CAD system. The geometrical definition and relationships are created in NX as user-defined features (UDF) and later linked to the EF-M design solution (DS) via a custom-made linking object. The approach focuses on the connection between the functional and the generation of the CAD model. However, the focus of the method is not on the evaluation of the component, although the generation of CAD had the intention of being evaluated. The method focused on the generation of alternatives; however, it did not cover the parametric variation of each CAD variant.

An earlier alternative for the UDFs by Muller et al. [20] worth mentioning is the configurable components approach [30], which is based on the concept of a component-based architecture. A systematic design methodology is proposed around this concept that builds on the E-FM methodology and adds the configurable component to complement the product architecture [31]. This work focuses on the conceptual constructs of the method without references to its practical software implementations. The background of the method is to develop a product family architecture to enable mass customization of products, unlike this article, which focuses on the quantitative assessment of different product architectures. Therefore, the geometrical domain is intended to support the manufacturing process rather than to evaluate its physical performance. In addition, the possibility of having associated models was suggested, as it represented *analysis and simulation models* within the configurable component objects. A conceptual implementation of the configurable component method is performed on the same product by Raja and Isaksson [16], focusing only on the definition of physical design.

2.2. Design Space Exploration from a Model-Based Systems Engineering (MBSE) Perspective

Even though the context of this paper is a component within a system, it is beneficial to explore the current approach from a systems engineering perspective. Systems engineering has a dedicated technical process to select the system architecture where an equivalent to “product variants” selection decision is performed: concept and technology selection according to NASA [32] or the architecture definition process according to INCOSE [12]. These manuals describe the process at a very generic top level and leave the implementation to the systems engineer. Methods such as ARCADIA [33] describe how to map the functional needs layer to the physical layer through a logical architectural layer. However, it does not systematically cover the inclusion of different architectural variants and their combinations when the system is in the physical embodiment phase of the design. Therefore, we consider that systems engineering methods such as ARCADIA and their supportive tools such as CAPELLA are intended to consider one architecture at a time. Recently, a method that focuses on functional requirements and component alternatives has been proposed to explore the architectural design space and optimization strategy in the MBSE domain Bussemaker et al. [27]. It proposes an architecture design space graph (ADSG) to capture the connection between functions and components. The DS-FR-DS tree in EF-M is equivalent to a FUN-COMP-FUN where FUN represents a function, and COMP represents a component, the equivalent of a design solution. The ADSG also allows one to capture design variants (“Architectural Options”) in a single graph but requires the explicit object “OPT” (Option) to be declared as opposed to the EF-M methodology, where it is implicit. Other notable differences include the possibility of explicitly including ports (input/outputs for variables) and attributes in the graph, while the EF-M configurable components [30,34] contain those elements within the design solution Object. Since ADSG has been created for the purpose of exploring the architectural design space, the logic

allows a straightforward generation of architectural variants, which can later be connected to a multi-disciplinary analysis and optimization (MDAO) framework to evaluate, in detail, each of the variants [35].

2.3. Knowledge-Based Engineering (KBE)

There is no single definition of KBE today, even though the discipline was born in the 1980s [23,36]. The definition of KBE in this paper is the same as that of La Rocca [23]: a combination of design rules and a CAD engine to support the generation of multiple physical models with the idea of modularization for reuse. In addition, a KBE system is required, such as ICAD, AML, INTENT!, or more recently, ParaPy. Other authors view KBE as almost synonymous with design automation support within a CAD system, such as knowledge fusion in Siemens NX. Examples of aerospace component analysis using the former include Isaksson [37] and Van Den Berg and Van Der Laan [38], Bas van Manen et al. [39], while examples of the latter include Martinsson Bonde et al. [25], Madrid et al. [40].

The KBE system methodology has not significantly changed from the 1990s, whereas the implementation and digital tools using the KBE approach have. The structures that persist are the library approach to store coded rule-based knowledge in primitives for reusability, demonstrated several decades ago by Chapman and Pinfold [41,42]. Another persistent approach is the knowledge acquisition process, which is a subset of the informal MOKA methodology [43]. However, the evolution of tools and methods has facilitated the introduction of the KBE paradigm. Changes include the incorporation of a friendly language to code: Python in ParaPy, as opposed to LISP or its derivatives. Another significant evolution is the incorporation of the development and operation (DevOps) techniques in the software industry that are now available to manage and deploy code, such as *git* version control, application deployment via containers, agile code development, or open-source server and web interfaces. The incorporation of those industry-standard techniques has reduced the KBE entry barrier.

Since the underlying methodology is the same, some fundamental challenges of KBE remain. KBE reviews have appeared periodically [23,36,44,45], focusing on different challenges and identifying hurdles that hinder the wide adoption in the industry of a method that theoretically saves design time via automation.

Three challenges are worth lifting from the literature review. The first one, identified in 2012 by Verhagen et al. [44], is “*effectively sourcing and re-using knowledge*”. The last review in 2023 [36] defines this challenge. *This problem is referred to as the incapability of exchange standards, which only enable the transfer of an instance of design and not the knowledge embodied to generate it.* The interpretation from the authors’ experience on KBE systems is that when significant efforts are spent in KBE primitives to capture the detailed behavior of a product, they are done for a specific product or project. In such scenarios, development time and capture of the specific product behavior take precedence over the generalization capability of such a primitive system. As a result, the automation effort is not easily translated to the next generation of products or even between variants of the same product.

The second challenge to raise is the difficulty of connecting the physical and functional domains. Reddy et al. [45] already mentioned that “*in general, the reason for designing a [physical] product is to meet a certain function*”. Similar challenges on the functional to physical connection are found in Ranta et al. [46].

The third challenge is the black box problem. The three reviews described above all highlight the perception of a black-box system due to the complexity of the code when real products are modeled. Our experience is that you need to have expert knowledge of the KBE system and the particular product application to understand its logic. It is also difficult to debug the code or pass it on to other engineers.

One characteristic of KBE systems is that they interact with a CAD kernel, either directly like ICAD or ParaPy (using OpenCascade) or via CAD suites (NX/KnowledgeFusion or Catia/Knowledgeware). Therefore, KBE applications traditionally involve the physical generation of design alternatives and often their evaluation. Recently, the work has also been expanded to accommodate, to some extent, the generation of architectural alternatives [28].

Previous work related to the design automation evaluation of aero engine components, such as the one used for verification of this paper, are as follows:

- Runnemalm et al. [47] created shell models using KBE for welding simulation.
- Martinsson Bonde et al. [48] carried out a study based on UDF and created solid models for FEA.
- Thor and Isaksson [49] explored the modularization of analysis methods in physical components of the same product family.

However, none of the KBE systems or applications mentioned before rely on a function model or support architectural decisions as a fundamental part of the implementation. Therefore, a research study has been identified, and a new method that combines functional and physical design space exploration has been developed here.

3. FUSE Method Description

To facilitate the automation of different functional architectures and their evaluation, in this section, a new method is proposed called FUSE (**f**unction and **s**tructure design space **e**xploration). A set of required inputs for FUSE are listed. Then, the FUSE steps are described in detail. Finally, a suggestion of the activities to be performed after the method is presented. The process of generating variants and conducting evaluations is illustrated in Figure 3, alongside the corresponding steps of the method.

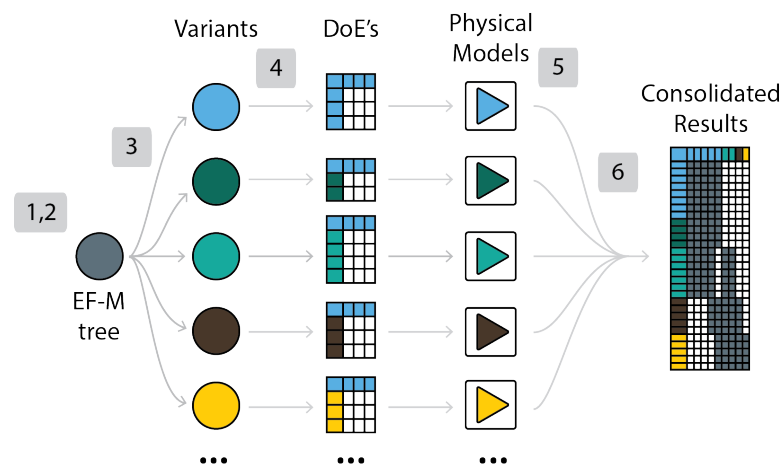


Figure 3. The method's steps are visualized here. (1) Mapping exercise between EF-M DS's and physical models. (2) DoE information stored in the EF-M tree. (3) The EF-M tree is instantiated, generating all variant configurations. (4) Each variant's DoE is generated, creating design cases. (5) Design cases are evaluated, and the results are extracted. (6) Results for each design case and variant are collected. Note that different variants, due to the nature of the architectural changes, may not have the same parameters in DoEs, represented as white cells in the consolidated results.

3.1. Previous Steps and Input Required

Four main inputs are required for the method:

1. **Enhanced-function means tree:** During conceptual assessments, a list of functional requirements and potential design solutions shall be available in the form of an

enhanced function-means tree. This includes incompatibilities between different design solutions already mapped in the tree.

2. **List of design space variables:** In practice, it implies defining the physical design space exploration for every design solution. The type of variable is identified and characterized. For example, continuous variables are defined by the bounds, and discrete variables are defined by alternative options.
3. **Quantities of interest:** The quantities of interest or performance values to evaluate the different concepts against each other shall be identified.
4. **Evaluation models:** The physical or analytical models and methods for calculating these quantities of interest shall be clearly defined.

The authors recognize that there may be some iteration between physical model preparation and design space exploration methodology. For the simplicity of the method exposition, the iterative loops are not explicitly stated.

3.2. Method Description

The method consists of six steps. Figure 4 visualizes the sequential steps.

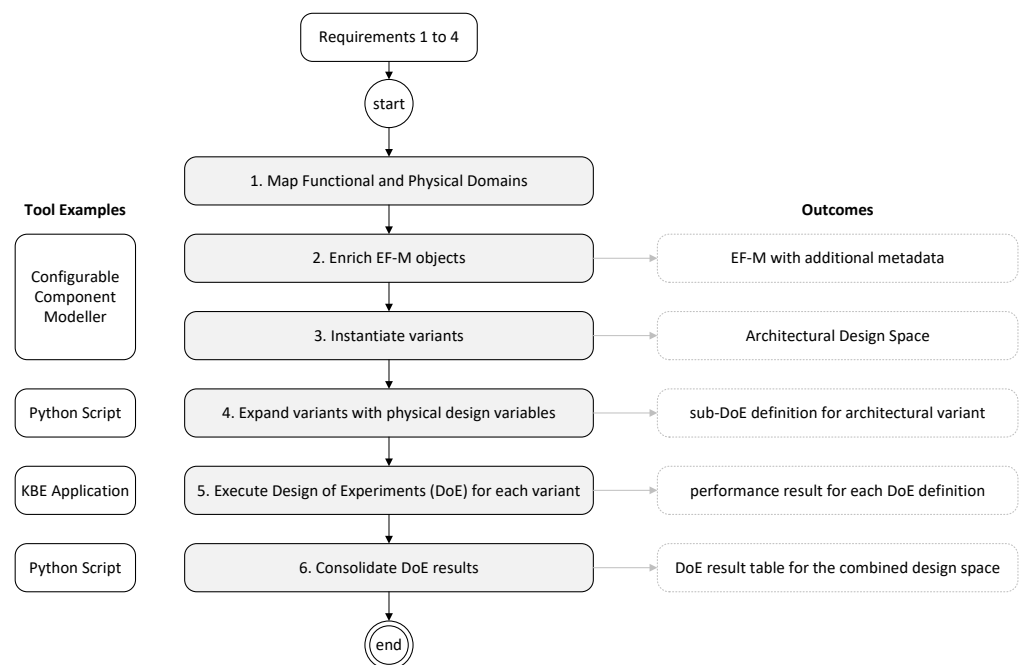


Figure 4. Visualization of the steps of the methodology next to suggestions on tools to perform the steps and the outcome of each step.

3.2.1. Step 1: Functional Map and Physical Domains

In the EF-M model, the designer identifies the design solutions (DS) that correspond to the variables or parameters that define physical design and space exploration. Note that highly integrated structures [11] may have a different design solution for the same physical component based on the provided functionality. The method relies on the designer to map the appropriate design solution.

3.2.2. Step 2: Enrich EF-M Objects

The design solutions identified in Step 1 are enriched with metadata. The metadata consist of variables or parameters to be varied in the physical design space for each DS. There are three possible types of metadata to add to the objects:

- Continuous variables require an upper and lower bound. Continuous variables can take any value between the upper and lower bounds.
- Discrete variables require a list of options. Discrete variables can take only one of the options provided.
- Configuration parameters can only take one value. The configuration parameters prepare geometrical models with specific configuration variables or parameters that are required to represent a particular design solution.

In practice, EF-M trees are modeled in software with a graphical user interface, such as the configurable component modeler (CCM) or Morpheus [50]. The outcome of this step is an updated EF-M tree model that contains both the architectural variants as part of tree branches and physical parameters as metadata on design solutions.

3.2.3. Step 3: Instantiate and Export Variants

This step constitutes the generation of architectural design space variants or alternatives. The algorithm in the software takes into account the EF-M tree structure, including the *interactions with* relationships between design solutions to generate a combinatorial list of possible valid configuration architectures. The list containing the physical metadata is then exported outside of the functional software. This method proposes a standard JSON output file for ease of parsing. The exported JSON contains architectural variants and, within them, design solutions selected for each functional requirement. Within design solutions, different design variables and configuration parameters are stored.

3.2.4. Step 4: Expand Variant DoEs

Each variant is parsed to extract the variables and parameters that constitute the physical design space, traditionally called design of experiments (DoE).

In the use case, we assume a simple design space exploration method by generating design cases using Latin hypercube sampling (LHS) techniques. Potentially, a design space optimization problem could be defined by adding objective and constrained parameters to the EF-M, using, for example, the CMDOWS standard [51].

Parsing and expanding the architectures is automated with a Python script. The script takes into account all the variables with their respective bounds or options and returns a design table with the actual values sampled from the variable definition *for each architectural variation*, as the bounds or options may be different for each variable. The outcome of this step is a DoE table with two levels: architectural top-level alternatives and the physical variations of each. The combination of both constitutes a design case.

3.2.5. Step 5: Execute DoEs

The physical models are instantiated and automatically updated with each design case parameter and variable. The method allows us to use any modeling technique that the designers are familiar with to obtain the physical performance parameters that the designers are interested in, for example, Catia, NX, ANSYS, or KBE software. The method also allows the user to specify the execution workflow manager desired: PIDO tools such as OptiSLang, Optimus, or other alternatives such as CMDOWS [51], KBE applications in ParaPy, or even in-house workflow automation scripts.

The outcome of this step is the physical performance evaluated for each design case. Given the multi-disciplinary nature of the evaluation, the outcome may include files to post-process and summarize each design case.

3.2.6. Step 6: Consolidation of Results

The final step in the method is to consolidate each of the performance results for each design case in the DoE table. The outcome of the method is a multiarchitectural DoE table with quantities of interest evaluated on the physical models. This step can be performed in parallel with the previous step.

The actual designs of the experiments table can be a two-dimensional table containing in columns the architectural and physical parameters and the performance results. Each row represents a design case. The DoE table can also be a structured list with each of the quantities of interest calculated, such as a JSON format file. The advantages of the two-dimensional matrix are the direct visualization and the possibility to manipulate it manually. The disadvantage is that for design solutions that do not share similar physical variables or architectural design solutions, the matrix can be very sparse. For those cases, a JSON file is recommended.

3.3. Next Design Steps After the Method

The following paragraphs exemplify how the method output data can be used in the next product development activities.

A parallel coordinate plot can be used to understand how the different variables affect the performance metrics. Alternatively, a multivariate plot can also be used to understand the relationship between the different architectures and parameters within the architectures at the same time.

Since DoE data are available, a response surface can also be generated. The response surface can be used to support the optimization of different architectures by replacing geometrical models, which are usually computationally expensive. As usual, this technique requires us to sample the design space sufficiently a priori to capture the behavior of the performance results.

4. Industrial Case Application

The presented use case was developed in collaboration with GKN aerospace engine systems, an RSP. It is an application of the FUSE methodology described above with the purpose of validating it in a typical conceptual design model. This section describes the overall problem faced, the preliminary inputs to the method, and how the method was applied.

4.1. Engineering Problem Description and Background

An engine OEM (the customer) started the conceptual development of a new product to significantly decrease greenhouse emissions, for which GKN is an RSP. GKN is responsible for designing and manufacturing a static component of that engine: a Turbine Rear Structure (TRS). Note that the TRS has different names depending on the engine OEM: turbine exhaust case (TEC), turbine bearing house (TBH), or turbine rear frame (TRF) are a few examples. Despite the novel architecture of the overall engine, the TRS component is driven by the same requirements as previous generation engines. Therefore, the TRS concept is similar to the previous design engines and is visualized in Figure 5. A cross-section of the TRS is shown in Figure 6.

In an effort to increase the environmental sustainability score of the component, GKN considers implementing the design in a new promising steel alloy that simplifies the manufacturing process and is also classified as a more sustainable alternative. However, the full material properties for high-energy impact and crack propagation at high temperatures are not available. A comprehensive material testing campaign (about one year) would exceed the time allocated for concept selection (3 months). GKN's desire to introduce promising

material in this generation of engines has led to reconsidering the conceptual architecture of the component.

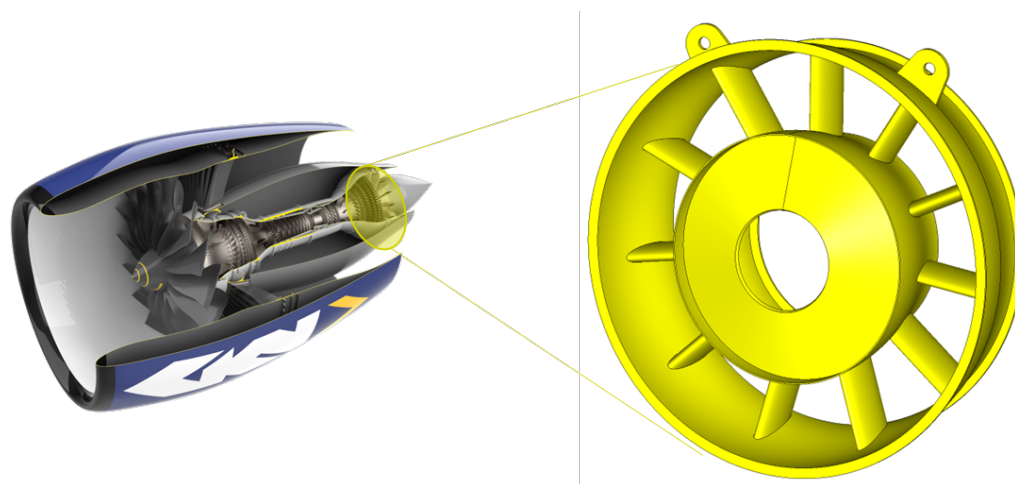


Figure 5. The turbine rear structure (TRS) component and its visualization in the context of a turboprop engine.

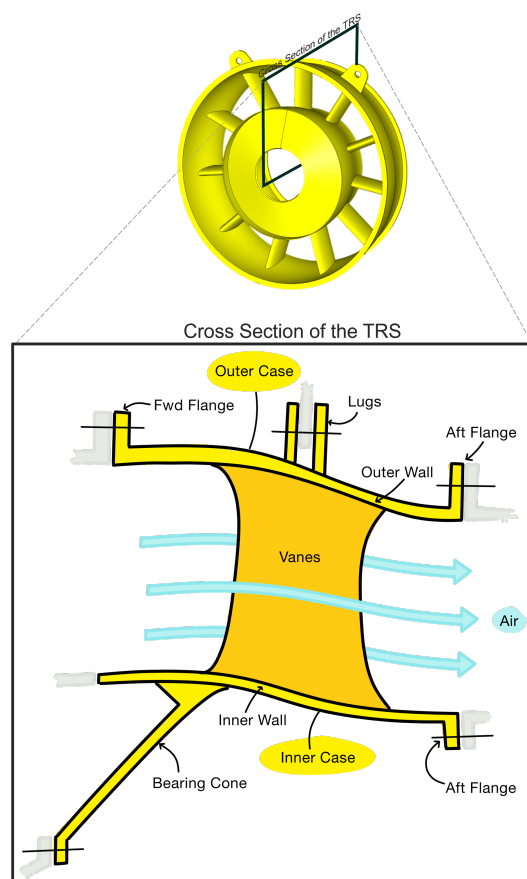


Figure 6. Cross-section of the TRS showing a traditional configuration.

The main engineering challenge revolves around the unknown capability of the new steel alloy to withstand a turbine blade failure scenario. The TRS is located immediately after the low-pressure turbine and is required to be able to contain its blade in the event of detachment. With no material properties of the new steel available in time, thickness, dimension, and weight of the component using the new material cannot be estimated. In addition, redesigning the TRS when final material data are available is a risk to the product development schedule.

The GKN concept team proposed a novel concept: decoupling requirements into two physical parts. A new part, an independent containment ring, made of a traditional material would fulfil the containment requirement, while the rest of the TRS would be made of a new and more sustainable material. This solution was patented in [52].

With the new architecture, the design team is concerned with the stiffness of the component; it must be similar to the baseline architecture. Therefore, a design space exploration is desired to compare the baseline architecture (integrated containment on the outer case of the TRS) vs. the new independent ring architecture. In addition, the team is interested in investigating whether the rear engine mounts that connect the TRS (and the engine) with the aircraft wing pylon require a single- or double-lug configuration.

4.2. Preliminary Steps

This section describes the use-case inputs that are available prior to the method's application.

The first input required is the functional model of the TRS in the form of an enhanced function-means tree (EF-M). The main functional requirements (FR) of the TRS have not been modified for this use case. For the benefit of the reader, the TRS top-level functional requirements in the EF-M tree are as follows:

- **FR1:** Maintain structural integrity. As a static component, the TRS transfers the loads from the neighboring engine components to the pylon. This is performed by the physical structure (inner case, outer case, and struts).
- **FR2:** Provide containment of the turbine blade. This is performed on the baseline by the outer case.
- **FR3:** Guide airflow from the turbine to the nozzle. This is satisfied by the inner case and outer case annular profiles.
- **FR4:** Reduce swirl of core flow. This is satisfied by the aerodynamic profile of the vane (also called the strut in its structural integrity function).
- **FR5:** Support and connect to neighbor components. This is performed by different interface definitions, such as case flanges, attachments, or bearing arms.

However, at a lower level, the EF-M tree model is updated from the baseline configuration. A new design solution for an *independent containment ring* is added to the functional requirement to *provide containment of the turbine blade*. In addition, the functional requirement *connected to the engine mount* is created, and two design solutions, *single lug* and *double lug*, are added, as shown in Figure 7.

The second input required for the method is a list of design variables. Since the design scenario is concerned with the stiffness of the component and the lug configuration, the design variables in Table 1 have been selected by GKN's product experts to represent the dimensions and boundaries of the exploration that are feasible in this scenario.

The third input required is the definition of the quantities of interest and their evaluation method. In this use case, four quantities were selected:

1. **Weight:** This is the default driver for all aerospace components. It is calculated using the CAD volume and the density of the material.
2. **Stiffness:** The stiffness of the component is important for the system as the mechanical whole engine model (WEM) models each component's stiffness and uses it to distribute the external load accurately. The calculation method uses an FEM with 3D elements. Unitary loads are applied at the component's interfaces (flanges), and displacements are extracted from the FEM results.
3. **Lug stress and failure modes:** The lugs in the TRS are considered part of the aircraft system and, therefore, subject to the CS-25 certification specification in Europe (14 CFR Part 25 in the United States). In particular, the CS 25.301 limit and ultimate analyses.

A well-established hand calculation method from the US Air Force (AD0759199), has been implemented in Python and is part of the KBE primitives.

4. **Containment capacity:** This is the ability of the outer case of the TRS to contain a rotating turbine blade (and disk) that breaks and impacts the TRS, according to CS-E 810 (CFR §33.94). For the preliminary design phases, a simple energy–strain model is used and compared to a reference model. In a detailed design scenario, a dynamic simulation of the blade section hitting the outer case is conducted.

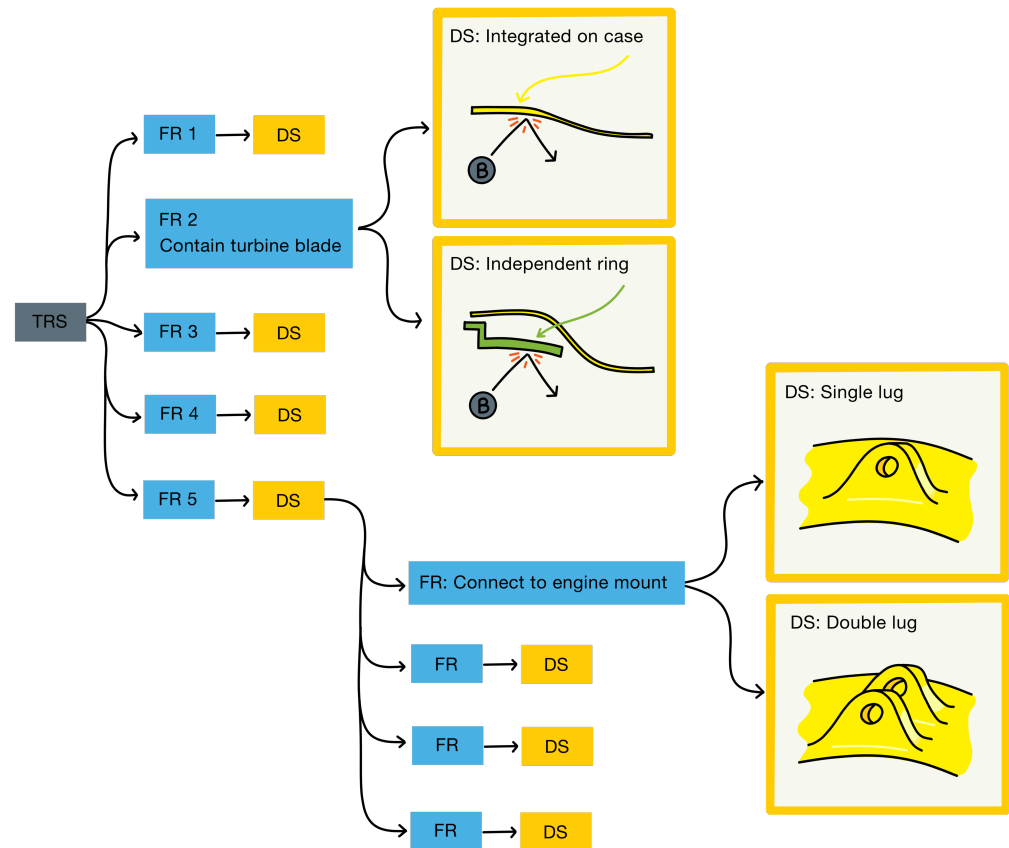


Figure 7. TRS modeled in the functional domain as an EF-M tree. The model has been simplified for visualization purposes. The functional requirements (FRs) with more than one design solution (DS) have been expanded.

Table 1. Description of DoE variables.

Variable Name	Variable Type	More Information
Number of vanes	discrete	Options = [6, 7, 8, 9, 10, 11] [adim]
Vane thickness	continuous	Bounds = (1, 3) [mm]
Ring material <i>if applicable</i>	discrete	Options = [Steel A, Steel B] [adim]
Lug thickness <i>single lug</i>	continuous	Bounds = (10, 20) [mm]
Lug thickness <i>double lug</i>	continuous	Bounds = (8, 15) [mm]
Outer wall thickness	continuous	Bounds = (4, 5) [mm]

Finally, the fourth input required for the FUSE method is the evaluation model. The TRS is modeled as a KBE application in the physical domain using the ParaPy KBE system. The primitive structure is presented in Figure 8 as a UML class diagram. The application is built from a library of primitives that, when combined, can generate a 3D geometry model and is also associated analysis models such as a finite element mesh, boundary

condition, and load application points. The KBE application includes analytical formulas for the quantities of interest.

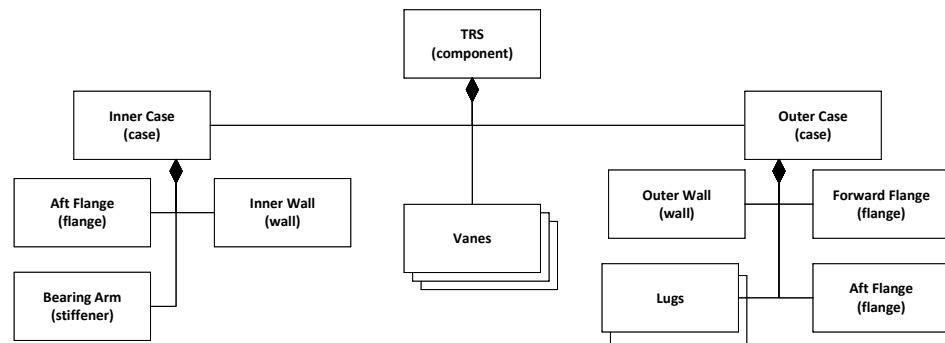


Figure 8. A UML class diagram of the KBE application. An extra class “Case” is added to group all MDFs into a single object for organization purposes.

4.3. Method Application

Steps 1 and 2 were performed in parallel. The software CCM v3.7 was used to model the EF-M tree and capture the metadata. With a list of the design space exploration variables available, the corresponding design solution objects were identified and their properties updated. On top of the design variables, additional configuration parameters were needed to correctly set up the model for each architectural solution. For example, the design solution *independent containment ring* has the following variables:

- *trs.outer_case.wall.containment_ring.material_name* = ‘Steel A’, ‘Steel B’ as a discrete design variable to select one of the two values. In addition,
- *trs.outer_case.containment_type* = *independent* to set up the KBE application to use the independent configuration;
- *trs.outer_case.wall_thickness* = 2 points to another KBE object geometry (the wall) to reduce the thickness to 2 mm as the containment requirement is no longer fulfilled by the outer case.

The full model is exported (Step 3) in a JSON format and then parsed, defining the design space exploration in Figure 9.

		Variants			
		1	2	3	4
<i>Variant parameters</i>	<i>type</i>				
Containment type	parameter	integrated	independent	integrated	independent
lug type	parameter	double	double	single	single
wall thickness	parameter		2		2
<i>Design variables</i>	<i>type</i>				
vane thickness	continuous	(1, 3)	(1, 3)	(1, 3)	(1, 3)
number of vanes	discrete	[6, 7, 8, 9, 10, 11]	[6, 7, 8, 9, 10, 11]	[6, 7, 8, 9, 10, 11]	[6, 7, 8, 9, 10, 11]
material	discrete		[Steel_A, Steel_B]		[Steel_A, Steel_B]
lug thickness	continuous	(8, 15)	(8, 15)	(10, 20)	(10, 20)
wall thickness	continuous	(4, 5)	(4, 5)	(4, 5)	(4, 5)

Figure 9. Definition of the design exploration space, combining architectural alternatives (variants) with their unique design variable definitions. Note that continuous variables are defined by their bounds, and discrete variables are defined by their options.

Each architectural selection is expanded (Step 4) using Latin hypercube sampling (100 samples for each architecture). Using the KBE application, the model is updated for each design case (Step 5). It takes 1 min to update the geometrical model and 1 s to calculate all quantities of interest except the model stiffness. The 3D finite element mesh takes 7 min on average to create, 2 min to solve the mesh, and 1 s to post-process the deformations and calculate the stiffness. The results are collated (Step 6) using python and exported to a CSV file. In total, the generation of results takes 20 h.

4.4. Results

Out of the 400 design cases (four architectures with 100 sample points each), there were 309 design cases that were able to generate a CAD, 208 of those were able to create a FEM mesh, and 190 were able to generate a successful FEM run. Therefore, the KBE implementation was able to successfully calculate the quantities of interest in 48% of the design cases. The incompleteness of the design space was due to the following:

- The combination of unfeasible design points was due to incompatibilities when sampling the geometrical parameters (23% of cases).
- The implementation of the KBE application was unable to generate a valid FEM input file (29% of cases).

The design space could have been better sampled if the design variable limits had been chosen more carefully and the KBE implementation made more robust. The valid 190 cases were evaluated by GKN experts, who considered that they sufficiently covered the design space under evaluation. Since the purpose of the use case was to test the FUSE method and not the use case KBE implementation, the results were accepted and processed.

The results presented in Figure 10 constitute the complete design space. Each line represents a successful design case run. The columns are variables or quantities of interest divided into three main groups:

- Design solutions:
 - containment type;
 - Lug type.
- Design variables:
 - Vane thickness;
 - Number of vanes;
 - Material;
 - Lug thickness;
 - Wall thickness.
- Quantities of interest:
 - Mass;
 - Containment capability;
 - Lug failure modes ($\times 4$);
 - Stiffness ($\times 10$).

A plot comparing stiffness to total mass (Figure 11) was provided to the component designers after processing the FUSE method results. With the plots and the raw results, the designers were able to downselect two relevant design cases that are detailed in Table 2.

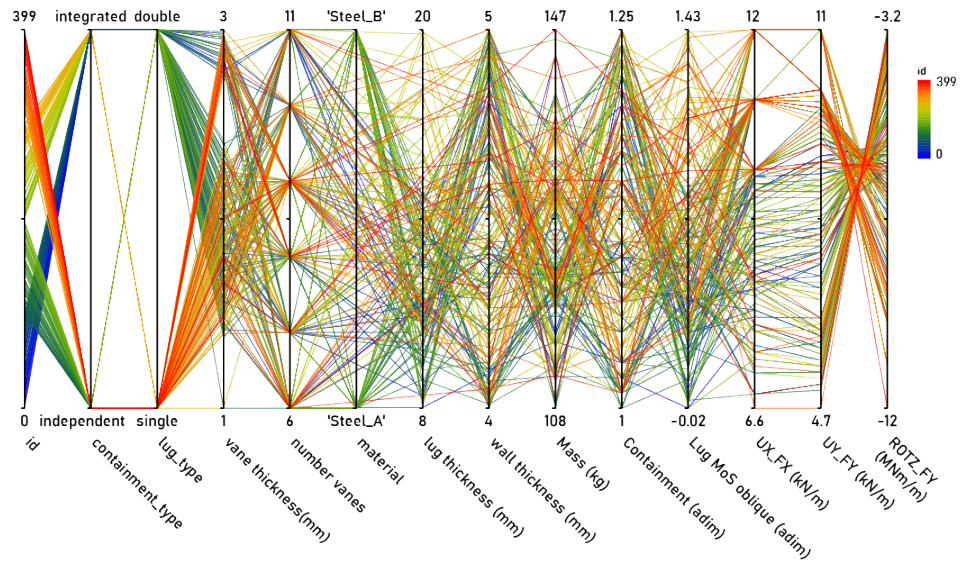


Figure 10. Parallel coordinate plot of the design space explored. Note that only a selection of quantities of interest are shown for visualization purposes.

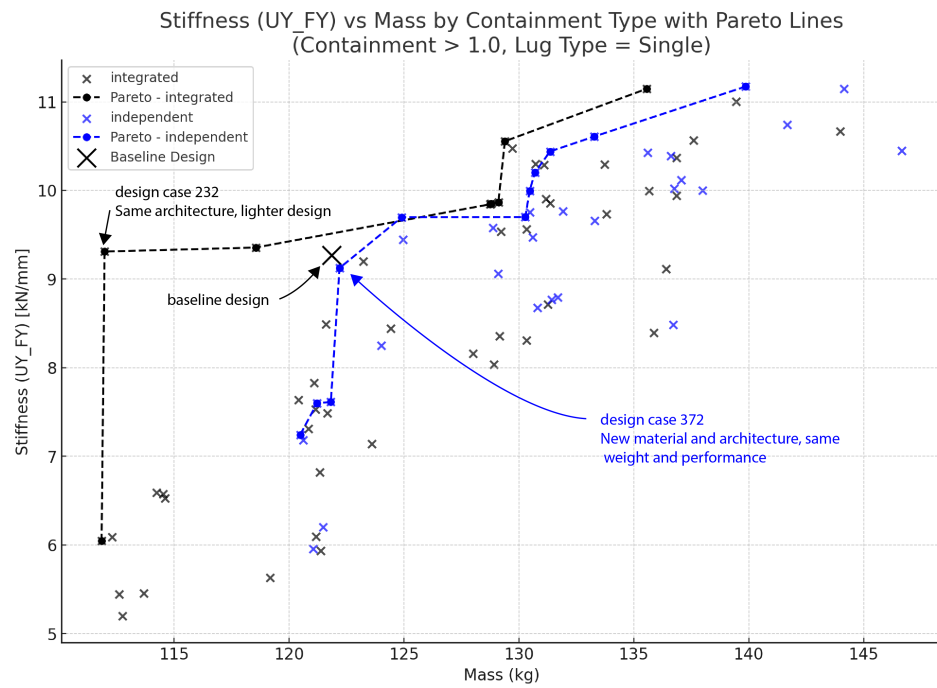


Figure 11. Behavior of the design space of different solutions showing two discrete architectures: integrated (black) and independent (blue). Each architecture is sampled (x-marks), and the non dominated design cases of the exploration are joined in a Pareto line.

- There is an independent architecture design case (id:372) that is able to provide similar performance to the baseline design with a traditional integrated architecture.
- The baseline integrated design was not an optimized design. The design exploration found a design solution (id:232) that was able to provide the same stiffness with 8% less weight

Table 2. Selected designs from design space exploration.

id	Containment Type	Mass (kg)	UY_FY (N/mm)	Comment
-	integrated	121.85	9270	baseline configuration
372	independent	122.20	9125	similar values
232	integrated	111.98	9310	same stiffness, -8% mass

4.5. Benchmark with the Traditional Method

The FUSE method is compared with a traditional approach to GKN, which uses OptiSlang as the workflow manager to automate design space exploration. The timescale to perform a similar setup is one working day (8 h) due to the need to set up each workflow manually for each configuration, given that different variables and values have to be passed through to the models. In contrast, it took 4 h to set up the design space exploration using the FUSE method. With only two functional requirements containing two design solutions each, we can observe that it took half of the time to set up. Using the same rationale as Muller et al. [20], we observed that this method reduces the effort to set evaluation workflows, especially as the design alternatives increase, exponentially exploding the number of individual workflows to set up.

4.6. Use-Case Conclusions

To conclude the use case, the functional and physical design space exploration methodology allows GKN designers to find a physical configuration for a new architecture (design case 372) that is able to de-risk the material data availability for containment while maintaining the performance of mass and stiffness. The method provided was able to provide an answer in 24 h. The setup time was 4 h, while the evaluation of design alternatives took 20 h in a standard laptop (i5 CPU) with the exception of the ANSYS FEM run which was offloaded to GKN's high-performing cluster. Design space exploration using traditional methods was estimated to take "at least two days" with the standard approach. The main challenge of the standard approach was reported to be the manual generation and configuration of each architectural variant.

5. Discussion

In this section, we discuss the advantages and disadvantages of the proposed FUSE method, including when it is useful for the designer and when other methods are more suitable. In addition, we revisit the related work section to draw comparisons and highlight how our methods stand in relation to existing approaches.

5.1. Lessons Learned: Flexibility of the Physical Modeling Approach

During the development of the method, there was constant feedback from aerospace designers, complementing the method validation and the industrial use case. Our experience with practitioners revealed that they had different physical modeling strategies during concept development: KBE vs. parametric CAD. The FUSE method was initially linked to the KBE domain, specifically KBE primitives, which were linked to the design solutions in the enhanced function-means tree. While it provided specific automation and time-saving benefits for the designer, it required us to impose a modeling constraint to allow the automatic linking between function and physical domains. The constraint, according to the designers, was too restrictive for the benefit of automation. The functional and physical models had to be specifically modified to adhere to the method, decreasing its adoption. It also added an additional abstraction layer that required the designer to learn before using the method. Flexibility is one of the valuable characteristics of function-means modeling. In order to maintain a high level of flexibility, Steps 1 and 2 were added to the method as manual steps to allow any modeling technique to be used in both function modeling (any EF-M tree will be valid) and physical modeling (CAD or KBE).

5.2. Strengths and Weaknesses

The overall objective of the method was to increase the connection between the functional and physical domains during the exploration of the design space. We believe that the

strength of this method is based on its simplicity and flexibility in connecting both domains. This method is most applicable when there are different design solution alternatives that interact with each other via incompatibilities or nested decision-making. Such scenarios are where the EF-M modeling excels in the design space exploration. The inclusion of different architectural variations in a single model allows a wider exploration of the design space with minimum overhead.

Another strength of this method over traditional architectural studies [26] is that it provides a response surface of the physically validated performance of the products in the form of response surfaces, as represented in Figure 12.

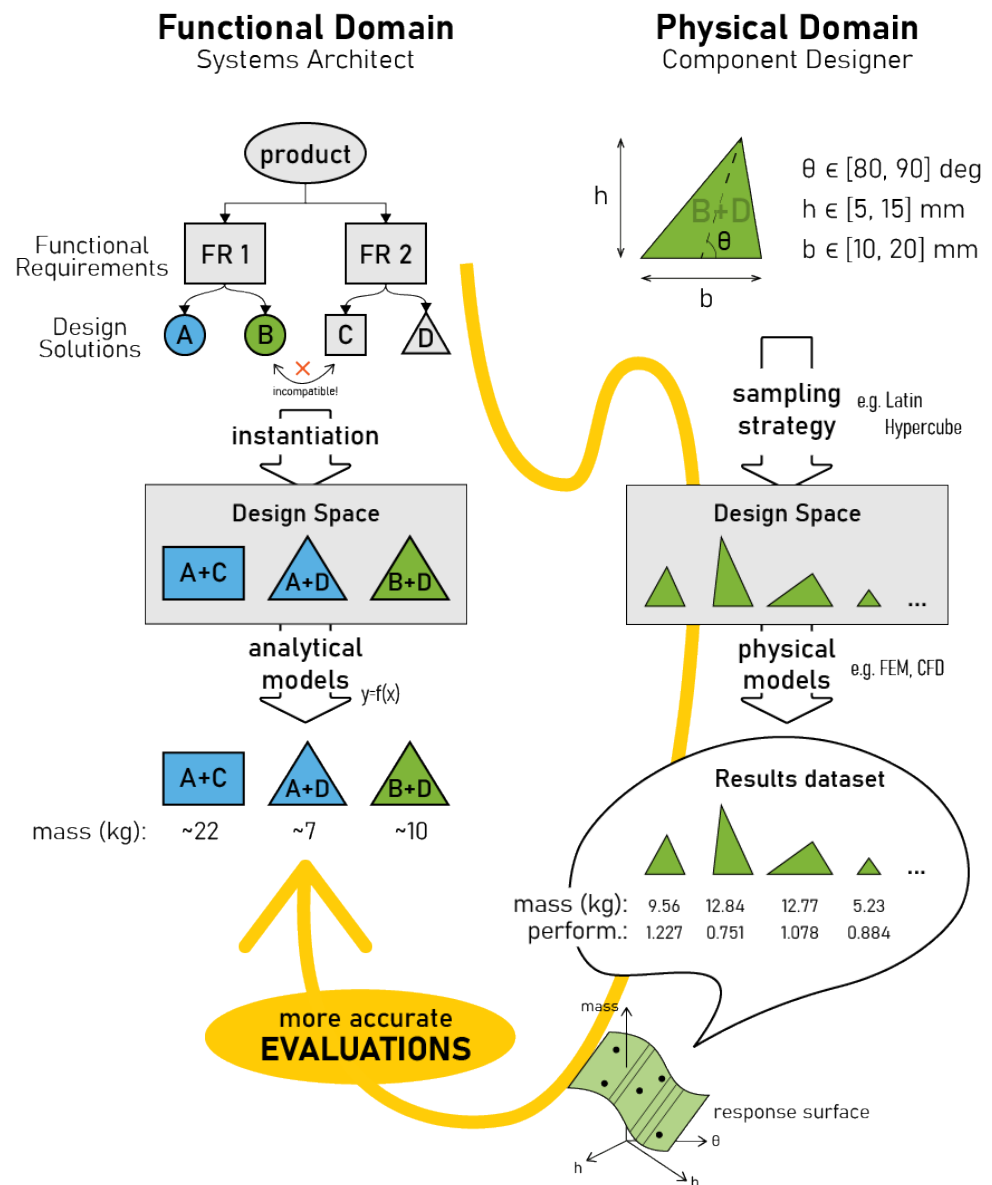


Figure 12. The overall approach of this method (in yellow) is to provide accurate physical evaluations for every architectural evaluation in comparison to analytical, first-order effect models.

To keep the method simple and flexible, some compromises were made, which can be seen as weaknesses compared to other methods. For example, the flexibility of choosing the physical modeling approach has the effect that there is loose and manual coupling between the design of experiments and the models. Other compromises were made during the implementation of the method regarding the ability to control design exploration or execution strategies. The implementation assumes an *a priori* sampling point strategy. In

this use case, it was Latin hypercube sampling. Adaptive sampling algorithms [53] can be incorporated into the method for efficient sampling.

Engineering design methods are only useful when the industry adopts them [54]. Although the flexibility of the method is an advantage, it also poses an acceptability challenge. Some GKN designers rejected it, citing the absence of a standalone tool with clear step-by-step instructions. Despite its demonstrated benefits and its appropriate communication, as per Gericke et al. [55], the effort required to implement new engineering methodologies was reported to be a key factor in their decision. The authors confirmed that the effort to adopt new engineering methodologies is behind it. This phenomenon is well documented in the engineering design research community [56,57]. As a result, this research confirms the challenge of method adoption and the need to facilitate the adoption of engineering design methods and tools by industry practitioners. Emerging technologies such as generative artificial intelligence (GenAI) interfaces based on large language models (LLMs) could be used to bridge this gap.

5.3. Comparison with Other Methods and Approaches to Explore the Design Space

Compared to the function and geometry design exploration method (FGE) [20], there is a difference in coverage: it expands the generation only to the evaluation of the design cases. Additionally, there is no restriction on the geometrical modeling strategy; therefore, it can be adopted by both KBE-inclined engineers or CAD automation engineers. If the KBE methodology is used, there is no need to create any algorithm to embody the design: the KBE methodology takes care of it automatically. The algorithm presented in [20] is necessary to embody different components, while this activity is explicitly defined by the user by the primitive structure of the KBE.

Compared to the architecture design space graph (ADSG) approach [27], the definition of the variations of this method is straightforward, as no *OPT* objects are required. This, we believe, helps the adoption of the method when only a simple design exploration is required. On the other hand, the ADSG approach was intended to not only control the architectures but also all the systems engineering inputs such as system specification and requirements. The ADSG method is also able to interact with MDAO graphs, which gives designers a prescribed procedure on how to execute the evaluation of the design space exploration. In contrast, the presented method does not specify such an interface.

We believe that our method is valid for early phase component or sub-system design exploration, where traditional MBSE methodologies are too rigid and cumbersome to generate. The cost of having this level of flexibility is the rigor of the architecture definition: MBSE systems can define architectures in a much higher level of detail and perspectives than EF-M models. For example, this method requires the designer to manually verify the validation of the requirements. For applications where MBSE and requirements validation are important, we recommend following Bruggeman's approach [58].

5.4. Limitations

The method has been validated using a single-use case example. A more robust evaluation should consider different use cases and scenarios. We believe that the exemplified use case is a generic enough application, and similar results are expected in different scenarios.

The effectiveness of the method is difficult to measure. The claim that in this use case, the design space exploration setup took 4 h to set up versus 8 h using traditional methods is not fully robust as it was only one measurement. This is highly dependent on the designer's abilities and familiarity with the tools and methods that were not measured here. The subjectivity of the time estimation and the singularity of the measurements are the main weaknesses of the paper. Since this method proposes an automation alternative to

a procedure with a high degree of manual modeling and judgement, a true comparison of pre- and post-design support is difficult. In addition, in practical industrial scenarios, these activities are performed by designers in segments between meetings, lunches, and holidays. Occasionally, bugs are found in the batch automation process, requiring designers to restart design exploration, further extending the lead time, and making measurements more difficult to compare. However, the level of validation is similar to other methods [20], and we believe that it is sufficient to exemplify the potential of this method.

5.5. Generalization of the Method to Other Use Cases

The method has been demonstrated in the design space exploration of a jet engine component. However, this method is applicable to any industry where functional requirements impact architectural changes for which physical (or other parameter-based) complex evaluations are required for meaningful design space exploration. Examples of such scenarios where this method is applicable are the automotive domain, where the electrification of the vehicle changes the functional requirements imposed on the physical product and, therefore, its architecture [59,60]. Similar scenarios may be faced by naval ship designers [61] or space vehicle architects [62].

6. Conclusions

A new method has been presented to facilitate the automation of design exploration, covering the functional and physical domains, including the ability to evaluate design alternatives. The method was evaluated in an industrial use case that allowed an improvement of the final method presented here and also as validation. The results of the method show that it can support the designer in down-selecting promising design cases. The advantages and disadvantages of the method have been presented, and comparisons with existing methods have been presented, highlighting the design scenarios where each method is most useful. The advantages of this method are the generation of architectural design variants and their physical evaluation of design spaces that are too wide to cover in detail using traditional tools.

Through the weaknesses and limitations of the method, a persistent engineering design challenge was identified: for methods to be adopted and to have an impact on practice, they must be easy to implement, both in terms of their theoretical framework and access to the tool supporting the method. Generative AI and large language models have been identified as possible solutions to interface between methodologies and engineers to lower the entry barrier and increase method adoption, which will be part of future research.

The aerospace industry is facing a challenge to move towards more sustainable solutions, and for that, new architectures are being studied. The architectures require **both** new technologies and new criteria to be evaluated. This method allows component designers to automate design space exploration and evaluation and to make quantitative trade-off decisions based on accurate measures.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/aerospace12010051/s1>.

Author Contributions: Conceptualization: A.P.G., M.P. and O.I.; Funding acquisition: M.P. and O.I.; Investigation: A.P.G.; Methodology: A.P.G.; Project administration: M.P.; Software: A.P.G.; Supervision: M.P. and O.I.; Writing—original draft: A.P.G.; Writing—review and editing: M.P. and O.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Swedish Innovation Agency VINNOVA, Project 19009 DEFAINE (the Design Exploration Framework based on AI for front-loaded Engineering). The project was enabled through EU Eureka ITEA3 call 6.

Data Availability Statement: The original contributions presented in this study are included in the article/supplementary material. Further inquiries can be directed to the corresponding author. The file *design_cases.csv* is contained in the supplementary material.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADSG	Architecture Design Space Graph
CAD	Computer-Aided Design
CCM	Configurable Component Modeller
CFD	Computational Fluid Dynamics
DoE	Design of Experiments
DS	Design Solution
DSE	Design Space Exploration
EF-M	Enhanced Function-Means
FEA	Finite Element Analysis
FEM	Finite Element Modelling
FR	Functional Requirement
JSON	JavaScript Object Notation
KBE	Knowledge Based Engineering
LHS	Latin Hypercube Sampling
MBSE	Model Based System Engineering
MDAO	Multi Disciplinary Analysis and Optimization
OEM	Original equipment Manufacturer
PIDO	Process Integration and Design Optimization
RSP	Risk Sharing Partner
TRS	Turbine Rear Structure
UML	Unified Modeling Language

References

1. European Commission. *Flightpath 2050 Europe's Vision for Aviation Maintaining Global Leadership and Serving Society's Needs*; Technical Report; Publications Office of the European Union: Luxembourg, 2011. [\[CrossRef\]](#)
2. Air Transport Action Group. *Waypoint 2050*; Technical Report; Air Transport Action Group: Geneva, Switzerland 2021.
3. Hallstedt, S.I.; Isaksson, O.; Nylander, J.W.; Andersson, P.; Knuts, S. Sustainable product development in aeroengine manufacturing: Challenges, opportunities and experiences from GKN Aerospace Engine System. *Des. Sci.* **2023**, *9*, e22. [\[CrossRef\]](#)
4. Isaksson, O.; Kossmann, M.; Bertoni, M.; Eres, H.; Monceaux, A.; Bertoni, A.; Wiseall, S.; Zhang, X. Value-Driven Design – A methodology to Link Expectations to Technical Requirements in the Extended Enterprise. *INCOSE Int. Symp.* **2013**, *23*, 803–819. [\[CrossRef\]](#)
5. Smith, D.J.; Tranfield, D. Talented suppliers? Strategic change and innovation in the UK aerospace industry. *R&D Manag.* **2005**, *35*, 37–49. [\[CrossRef\]](#)
6. Tiwari, S.; Pekris, M.J.; Doherty, J.J. A review of liquid hydrogen aircraft and propulsion technologies. *Int. J. Hydrogen Energy* **2024**, *57*, 1174–1196. [\[CrossRef\]](#)
7. Yeranee, K.; Rao, Y.; Xu, C.; Zhang, Y.; Su, X. Turbulent Flow Heat Transfer and Thermal Stress Improvement of Gas Turbine Blade Trailing Edge Cooling with Diamond-Type TPMS Structure. *Aerospace* **2024**, *11*, 37. [\[CrossRef\]](#)
8. Durocher, A.; Fan, L.; Francolini, B.; Furi, M.; Bourque, G.; Sirois, J.; May, D.; Bergthorson, J.; Yun, S.; Vena, P. Characterization of a Novel Additive Manufacturing Micromix Nozzle Burning Methane to Hydrogen. *J. Eng. Gas Turbines Power* **2024**, *146*, 5. [\[CrossRef\]](#)
9. Woodbury, R.F.; Burrow, A.L. Whither design space? *AI EDAM* **2006**, *20*, 63–82. [\[CrossRef\]](#)
10. Ross, A.M.; Hastings, D.E. The Tradespace Exploration Paradigm. *INCOSE Int. Symp.* **2005**, *15*, 1706–1718. [\[CrossRef\]](#)
11. Ulrich, K. The role of product architecture in the manufacturing firm. *Res. Policy* **1995**, *24*, 419–440. [\[CrossRef\]](#)
12. INCOSE. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*; John Wiley & Sons: Hoboken, NJ, USA, 2015.
13. Ulrich, K.T.; Eppinger, S.D. *Product Design and Development*; McGraw-Hill: New York, NY, USA, 2016.

14. Suh, N.P. *Axiomatic Design: Advances and Applications*; The MIT-Pappalardo series in mechanical engineering; Oxford University Press: New York, NY, USA, 2001.
15. Chakrabarti, A.; Blessing, L. Special Issue: Representing functionality in design. *Artif. Intell. Eng. Des. Anal. Manuf.* **1996**, *10*, 251–253. [[CrossRef](#)]
16. Raja, V.; Isaksson, O. Generic Functional Decomposition of an Integrated Jet Engine Mechanical Sub System Using a Configurable Component Approach. *Adv. Transdiscipl. Eng.* **2015**, *2*, 337–346. [[CrossRef](#)]
17. Guenov, M.; Molina-Cristobal, A.; Voloshin, V.; Riaz, A.; Van Heerden, A.S.; Sharma, S.; Cuiller, C.; Giese, T. Aircraft Systems Architecting—A Functional-Logical Domain Perspective. In Proceedings of the 16th AIAA Aviation Technology, Integration, and Operations Conference, Washington, DC, USA, 13–17 June 2016. [[CrossRef](#)]
18. Staack, I.; Amadori, K.; Jouannet, C. A holistic engineering approach to aeronautical product development. *Aeronaut. J.* **2019**, *123*, 1545–1560. [[CrossRef](#)]
19. Todorov, V.T.; Rakov, D.; Bardenhagen, A. Enhancement Opportunities for Conceptual Design in Aerospace Based on the Advanced Morphological Approach. *Aerospace* **2022**, *9*, 78. [[CrossRef](#)]
20. Muller, J.; Panarotto, M.; Isaksson, O. Function Model-based Generation of CAD Model Variants. *Comput.-Aided Des. Appl.* **2021**, *18*, 970–989. [[CrossRef](#)]
21. Schachinger, P.; Johannesson, H.L. Computer modelling of design specifications. *J. Eng. Des.* **2000**, *11*, 317–329. [[CrossRef](#)]
22. Wynn, D.C.; Caldwell, N.H.M.; John Clarkson, P. Predicting Change Propagation in Complex Design Workflows. *J. Mech. Des.* **2014**, *136*, 081009. [[CrossRef](#)]
23. La Rocca, G. Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design. *Adv. Eng. Inform.* **2012**, *26*, 159–179. [[CrossRef](#)]
24. Müller, J.R.; Panarotto, M.; Isaksson, O. Design Space Exploration of a Jet Engine Component Using a Combined Object Model for Function and Geometry. *Aerospace* **2020**, *7*, 173. [[CrossRef](#)]
25. Martinsson Bonde, J.; Kokkolaras, M.; Andersson, P.; Panarotto, M.; Isaksson, O. A similarity-assisted multi-fidelity approach to conceptual design space exploration. *Comput. Ind.* **2023**, *151*, 103957. [[CrossRef](#)]
26. Ölvander, J.; Lundén, B.; Gavel, H. A computerized optimization framework for the morphological matrix applied to aircraft conceptual design. *Comput.-Aided Des.* **2009**, *41*, 187–196. [[CrossRef](#)]
27. Bussemaker, J.H.; Ciampa, P.D.; Nagel, B. System Architecture Design Space Exploration: An Approach to Modeling and Optimization. In Proceedings of the AIAA Aviation 2020 Forum, Virtual Event, 15–19 June 2020. [[CrossRef](#)]
28. Sonneveld, J.; Bruggeman, A.M.R.M.; Beijer, B.; van den Berg, T. Dynamic workflow generation applied to aircraft moveable architecture optimization. In Proceedings of the Aerospace Europe Conference 2023—10th EUCASS—9th CEAS, Lausanne, Switzerland, 9–13 July 2023.
29. Müller, J.R.; Isaksson, O.; Landahl, J.; Raja, V.; Panarotto, M.; Levandowski, C.; Raudberget, D. Enhanced function-means modeling supporting design space exploration. *AI EDAM* **2019**, *33*, 502–516. [[CrossRef](#)]
30. Claesson, A.; Johannesson, H.; Gedell, S. Platform Product Development: Product Model—A System Structure Composed of Configurable Components. In Proceedings of the 13th International Conference on Design Theory and Methodology. American Society of Mechanical Engineers Digital Collection, Pittsburgh, PA, USA, 9–12 September 2001; pp. 317–326. [[CrossRef](#)]
31. Johannesson, H.; Claesson, A. Systematic product platform design: A combined function-means and parametric modeling approach. *J. Eng. Des.* **2005**, *16*, 25–43. [[CrossRef](#)]
32. NASA. *Systems Engineering Handbook*; NASA: Washington, DC, USA, 2006.
33. Roques, P. *Systems Architecture Modeling with the Arcadia Method: A Practical Guide to Capella*; ISTE Press-Elsevier: London, UK, 2017; p. 277. [[CrossRef](#)]
34. Raudberget, D.; Edholm, P.; Andersson, M. Implementing the principles of Set-based Concurrent Engineering in Configurable Component Platforms. In Proceedings of the 9th NordDesign Conference, Aalborg, Denmark, 22–24 August 2012.
35. Bussemaker, J.; Boggero, L. From System Architecting to System Design and Optimization: A Link Between MBSE and MDAO. In Proceedings of the 32nd annual INCOSE International Symposium, Detroit, MI, USA, 25–30 June 2022. [[CrossRef](#)]
36. Kügler, P.; Dworschak, F.; Schleich, B.; Wartzack, S. The evolution of knowledge-based engineering from a design research perspective: Literature review 2012–2021. *Adv. Eng. Inform.* **2023**, *55*, 101892. [[CrossRef](#)]
37. Isaksson, O. A generative modeling approach to engineering design. In Proceedings of the International Conference on Engineering Design, ICED, Stockholm, Sweden, 19–21 August 2003; Volume DS 31.
38. Van Den Berg, T.; Van Der Laan, A. A multidisciplinary modeling system for structural design applied to aircraft moveables. In Proceedings of the AIAA Aviation and Aeronautics Forum and Exposition, AIAA AVIATION Forum 2021, Virtual, 2–6 August 2021. [[CrossRef](#)]
39. van Manen, B.; van den Berg, T.; van der Laan, T.; Timmer, B. A Multidisciplinary Modeling System for Designing Fuselage Structures: Extending the Multidisciplinary Modeler. In Proceedings of the Joint 10th EUCASS—9th CEAS Conference, Aerospace Europe Conference, Lausanne, Switzerland, 9–13 July 2023.

40. Madrid, J.; Andersson, P.; Söderberg, R.; Wärmefjord, K.; Kveselys, D.; Lindkvist, L.; Löf, J. Automated and interactive evaluation of welding producibility in an multidisciplinary design optimization environment for aircraft components. *Int. J. Interact. Des. Manuf.* **2021**, *15*, 463–479. [[CrossRef](#)]
41. Chapman, C.; Pinfold, M. Design engineering—A need to rethink the solution using knowledge based engineering. *Knowl.-Based Syst.* **1999**, *12*, 257–267. [[CrossRef](#)]
42. Pinfold, M.; Chapman, C. Combining FEA and KBE Techniques to Automate the Analysis Process—A Foresight Vehicle Programme. In Proceedings of the SAE Technical Papers, SAE 2002 World Congress, Detroit, MI, USA, 4–7 March 2002. [[CrossRef](#)]
43. Stokes, M. *Managing Engineering Knowledge: MOKA-Methodology for Knowledge Based Engineering Applications*; ASME Press: New York, NY, USA, 2001.
44. Verhagen, W.J.; Bermell-Garcia, P.; van Dijk, R.E.; Curran, R. A critical review of Knowledge-Based Engineering: An identification of research challenges. *Adv. Eng. Inform.* **2012**, *26*, 5–15. [[CrossRef](#)]
45. Reddy, E.J.; Sridhar, C.N.V.; Rangadu, V.P. Knowledge Based Engineering: Notion, Approaches and Future Trends. *Am. J. Intell. Syst.* **2015**, *5*, 1–7.
46. Ranta, M.; Mäntylä, M.; Umeda, Y.; Tomiyama, T. Integration of functional and feature-based product modelling—The IMS/GNOSIS experience. *Comput.-Aided Des.* **1996**, *28*, 371–381. [[CrossRef](#)]
47. Runnemalm, H.; Tersing, H.; Isaksson, O. Virtual manufacturing of lightweight aero engine components. In Proceedings of the International Symposium on Air Breathing Engines, Montreal, QC, Canada, 7–11 September 2009.
48. Martinsson Bonde, J.; Brahma, A.; Panarotto, M.; Isaksson, O.; Wärmefjord, K.; Söderberg, R. Assessment of weld manufacturability of alternative jet engine structural components through digital experiments. In Proceedings of the International Society for Air Breathing Engines (ISABE), Ottawa, ON, Canada, 25–30 September 2022.
49. Thor, P.; Isaksson, O. Automated design of aero engine structural components: A modular approach that enables re-use of generative engineering methods. In Proceedings of the XXth International Symposium on Air Breathing Engines 2011: (ISABE 2011), Gothenburg, Sweden, 12–16 September 2011; Volume 3.
50. Martinsson Bonde, J.; Mallalieu, A.; Panarotto, M.; Isaksson, O.; Almefelt, L.; Malmqvist, J. Morpheus: The Development and Evaluation of a Software Tool for Morphological Matrices. In Proceedings of NordDesign 2022, Copenhagen, Denmark, 16–18 August 2022; pp. 1–10.
51. Van Gent, I.; La Rocca, G.; Hoogreef, M.F.M. CMDOWS: A proposed new standard to store and exchange MDO systems. *CEAS Aeronaut. J.* **2018**, *9*, 607–627. [[CrossRef](#)]
52. Svensson, J.; Dahl, E.; Russberg, D.; Sjöqvist, R.; Nilsson, J.; Johansson, S. A Turbine and a Component. WO2003104630A1, 4 June 2003.
53. Zhang, Y.; Neelakantan, A.; Park, C.; Kim, N.H.; Lam, H.; Haftka, R.T. Adaptive Sampling with Varying Sampling Cost for Design Space Exploration. *AIAA J.* **2019**, *57*, 1032–1043. [[CrossRef](#)]
54. Cantamessa, M. Design Best Practices, Capabilities and Performance. *J. Eng. Des.* **2010**, *10*, 305–328. [[CrossRef](#)]
55. Gericke, K.; Eckert, C.; Stacey, M. What do we need to say about a design method. In Proceedings of the 21th International Conference on Engineering Design (ICED), Vancouver, BC, Canada, 21–25 August 2017.
56. López-Mesa, B.; Bylund, N. A study of the use of concept selection methods from inside a company. *Res. Eng. Des.* **2011**, *22*, 7–27. [[CrossRef](#)]
57. Booker, J. A survey-based methodology for prioritising the industrial implementation qualities of design tools. *J. Eng. Des.* **2012**, *23*, 507–525. [[CrossRef](#)]
58. Bruggeman, A.L.; van Manen, B.; van der Laan, T.; van den Berg, T.; La Rocca, G. An MBSE-Based Requirement Verification Framework to support the MDAO process. In Proceedings of the AIAA Aviation 2022 Forum, Chicago, IL, USA & Virtual, 27 June–1 July 2022. [[CrossRef](#)]
59. Chen, R.; Liu, Y.; Fan, H.; Zhao, J.; Ye, X. An Integrated Approach for Automated Physical Architecture Generation and Multi-Criteria Evaluation for Complex Product Design. *J. Eng. Des.* **2019**, *30*, 63–101. [[CrossRef](#)]
60. Gan, S.; Chrenko, D.; Kéromnès, A.; Le Moyne, L. Development of a Multi-Architecture and Multi-Application Hybrid Vehicle Design and Management Tool. *Energies* **2018**, *11*, 3185. [[CrossRef](#)]
61. Doerry, N.H.; Fireman, H. Designing All Electric Ships. In Proceedings of the Ninth International Marine Design Conference, Dubrovnik, Croatia, 15–18 May 2006; pp. 475–498.
62. Frank, C.P.; Marlier, R.A.; Pinon-Fischer, O.J.; Mavris, D.N. Evolutionary Multi-Objective Multi-Architecture Design Space Exploration Methodology. *Optim. Eng.* **2018**, *19*, 359–381. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.