



Development of a Hybridization-Aware Digital Integrated Systems Model for a Baseline Aircraft Inner Fixed Structure

Miranda Stockhausen ^{*}, Barath Govindaraju Pillai Ramanathan [†], and Vishal Modi [‡]
Department of Aerospace Engineering, University of Michigan, Ann Arbor, Michigan, 48109, USA

Steven Kestler [§]
Collins Aerospace, Chula Vista, California, 91910, USA

Gokcin Cinar [¶]
Department of Aerospace Engineering, University of Michigan, Ann Arbor, Michigan, 48109, USA

As aerospace systems become more multidisciplinary, conceptual design increasingly depends on coordinated treatment of tightly coupled structural, thermal, aerodynamic, maintenance, and certification considerations. Traditional document-based workflows make this coordination difficult to manage and update consistently. This work presents this workflow using the conceptual design of a commercial nacelle's Inner Fixed Structure (IFS) as an example, using the Requirements-Functional-Logical-Physical (RFLP) process within a Model-Based Systems Engineering (MBSE) framework implemented in Cameo MagicDraw. The paper develops a baseline IFS digital systems model spanning stakeholder requirements, requirement normalization, activity-based behavior, logical architecture, physical architecture, and cross-layer traceability. The resulting model is hybridization-aware in that it is structured to support early identification of likely change-impact pathways, new interfacing needs, and additional thermal, electrical, and certification considerations associated with future hybridized propulsion concepts. Results demonstrate the baseline model as a technical design document and scalable reference architecture, improving clarity in subsystem definition and supporting a unified representation of multidisciplinary design considerations. The work therefore contributes a subsystem-level, traceability-first MBSE case study and establishes a baseline model prepared for later hybridization-oriented extension.

I. Nomenclature

<i>ASOT</i>	=	Authoritative Source of Truth
<i>DEE</i>	=	Digital Engineering Environment
<i>DOD</i>	=	Department of Defense
<i>HEP</i>	=	Hybrid-Electric Propulsion
<i>IFS</i>	=	Inner Fixed Structure
<i>INCOSSE</i>	=	International Council on Systems Engineering
<i>MBSE</i>	=	Model Based Systems Engineering
<i>MEA</i>	=	More Electric Aircraft
<i>RFLP</i>	=	Requirements engineering, Functional analysis, Logical and Physical architecture
<i>SME</i>	=	Subject Matter Expert
<i>SysML</i>	=	System Model Language
<i>SoI</i>	=	System of Interest

^{*}Graduate Student Research Assistant, Department of Aerospace Engineering, University of Michigan, Ann Arbor, Michigan 48109, AIAA Student Member

[†]Graduate Student Research Assistant, Department of Aerospace Engineering, University of Michigan, Ann Arbor, Michigan 48109

[‡]Undergraduate Student Research Assistant, Department of Aerospace Engineering, University of Michigan, Ann Arbor, Michigan 48109

[§]Principle Systems Engineer, Advanced Structures, Chula Vista, California, 91910, AIAA Corporate Member

[¶]Assistant Professor, Department of Aerospace Engineering, University of Michigan, Ann Arbor, Michigan 48109, AIAA Senior Member.

II. Introduction

DESIGN of aerospace systems has become an increasingly interdisciplinary endeavor, which places significant burdens on engineers in terms of communication and process integration [1]. As the complexity and scale of a system increases, so too does the size of the requirements and functional space which define the systems engineering for that system. Traditional design depends on standalone documentation for each facet of the design, leading to limitations on scale and challenging version control [2, 3]. As the document number grows, it becomes increasingly difficult to locate and update each file for rolling design changes, as well as manage supporting documentation for each decision. Further, integration requires considerations of emergent factors beyond those managed by the independent engineering specialties. The result is a time consuming and intellectually burdensome design procedure whose segmentation introduces additional risk for design gaps.

This is especially true for advanced and hybridized propulsion systems, which place additional emphasis on electrical engineering and component integration compared to traditional turbomachinery design [4, 5]. These complexities require new systems engineering tools and methods to ease communication challenges and enable more traceable product development in the aerospace sphere. This work contributes to that space by introducing a repeatable framework for engaging requirement-to-component design for an engine IFS in a modular, scalable MagicDraw digital engineering environment. The method applies the Requirement-Functional-Logical-Physical (RFLP) design paradigm to track stakeholder needs through the design development process; using system model language (SysML) for standardized communication. This paper introduces an IFS-specific model-based systems engineering (MBSE) workflow for conceptual design and hybridization-aware change-impact scoping, to investigate the capacity of the digital model to illuminate design tradeoffs and structure future hybridization extensions.

The digital model provides a centralized authoritative source of truth for modeling, traceability analysis, and structured review of requirement coverage during early conceptual stage design while reducing the traditional communication burdens rampant in multidisciplinary engineering. These benefits are demonstrated using the baseline digital model to explore the IFS design space and identify potential conceptualization gaps, change-impact pathways, and likely integration pain points relevant to future hybridization. This IFS-focused MBSE case study expands prior engine-inlet work by refining the SysML framework for modeling [6], and extending the procedure to hybridization-relevant thermal, electrical, and certification couplings. This paper proposes a traceability strategy which connects initial stakeholder needs, requirements, functions, and logical architecture. The outcome is an integrated baseline systems model that represents IFS functional behavior, captures compliance logic, supports layered traceability, and provides a structured basis for hybridization-oriented reasoning and future impact analysis.

III. Literature Review

A. Understanding the IFS

The IFS is the nacelle structure that encases the engine core and forms half of the bypass duct along with the translating sleeve [7]. Although a static structure, this component is responsible for managing loading and aerodynamic interfacing, and provides a key line of defense for the engine and nacelle from potential failure modes. It forms the structural body of the propulsion system, which supports the loads on the bypass duct to produce a clean aerodynamic surface for the fan. It contains the acoustic treatment, which damp fan and turbine noise. It also includes thermal insulation to prevent the transfer of heat from the engine core and the spread of fire [8, 9]. The IFS defines a smooth inner bypass duct wall to reduce drag through the engine and maintain the proper flow profile [7]. Discontinuities or gaps at the interface can degrade bypass flow and limit fuel efficiency, making the IFS an important factor in the aero-structural design tradeoff. Finally, the IFS is the last barrier to engine access, so maintenance turn around and line activities must be considered during its design [9]. Overall, the IFS shown in Fig. 1 is a complex propulsive subsystem and an aero-structures component with significant interdisciplinary considerations.

This interdisciplinary complexity is likely to increase, as future IFS designs are expected to integrate more mechanical and thermal management components to accommodate electrification [10, 11]. With the shift toward hybrid-electric propulsion (HEP) systems and more electric aircraft (MEA), the IFS is anticipated to further integrate components for thermal management in addition to the existing structural requirements [10]. Future IFS architectures must accommodate high voltage power distribution, advanced cooling loops for power electronics, and electromagnetic compatibility requirements [12, 13]. In addition, stringent thermal management, material selection, and regulatory design directives will be required to ensure compliance with evolving airworthiness and certification standards for high voltage, high temperature aircraft systems. Recent hybrid electric propulsion studies highlight the expanded

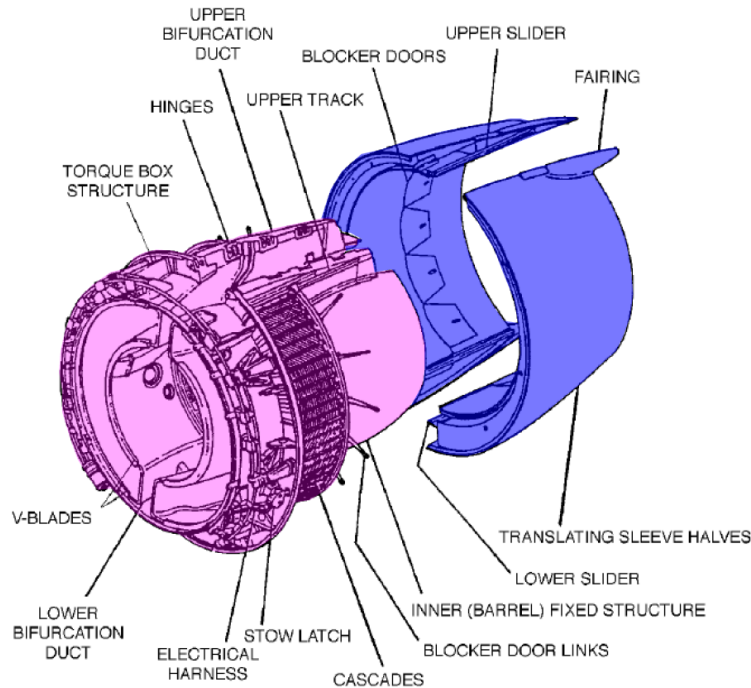


Fig. 1 Representative configuration of the Inner Fixed Structure (IFS) and adjacent nacelle components from Collins Aerospace [7].

coupling between propulsion, structure, and systems integration, reinforcing the need for model-based, traceable design frameworks [2, 12, 14].

B. Transitioning from Document-Based to Digital Engineering & Model-Based Design

Traditional design processes in aerospace and other industries have depended on document based methods, where each design decision and artifact is represented by a set of static written records which define its decision making, development, and verification [2, 3, 15]. This method imparts a number of challenges, particularly for complex system design, which have been recognized in the literature. According to Oroz et al, traditional documentation suffers from inefficiencies because data for artifacts are stored independently in a siloed structure which reduces engineers' access to information [3]. This is often exacerbated by the use of different and incompatible systems between different businesses. Design iterations must be updated manually for all documents, and there is little change management available to ensure that this propagation is standardized or tracked as design evolution renders physical documents obsolete [2]. Further, there is a lack of complex design semantics to guide consistent interpretations of documentation and schematics. These limitations create barriers to consistent communication between agents and introduce opportunities for designers to act on incomplete or out-of-date information, leading to design failures. All together, this induces higher development costs as time and money must be spent iteratively updating an ever-increasing and segmented set of documents, and then managing late-stage design changes.

Although these challenges are common to most fields of study involving design, they are concentrated in propulsion system design because of the number of interfaces amongst different suppliers as well as the number of disciplines concentrated in a single system [15]. Aircraft propulsion includes high-order physical interactions and numerous couplings which touch not only the internal components, but nearly every other part of the aircraft involving almost all aerospace disciplines. These factors are only intensified by the introduction of hybridization. The predicted changes to the IFS for future aircraft designs imply a growing multidisciplinary dependency between structural, electrical, and thermal subsystems, which render the documentation based methods even more time consuming. This makes *digital* systems engineering methods with strong traceability threads, such as MBSE, particularly valuable for future propulsion system development [14]. In contrast to document-based design methods, digital engineering and MBSE apply integrated computerized, model-based approaches to engineering problems. The paradigm leverages computational methods and

surrogate modeling concepts to establish a centralized source of truth for design processes, thus moving the engineering process away from traditional document-based design-build-test methods[16]. Rather, a model based approach is engaged, where digital artifacts are used to develop and analyze the system prior to physical construction [17]. This digital environment enables more reliable product maturation with fewer iterations, by establishing a centralized, authoritative, and up-to-date source of information for the development process [16]. A central platform facilitates the tracking and integration of data, information, and design updates from the conceptual phase through the full life-cycle of the product. This unified and traceable digital thread has garnered increasing attention for its potential role in improving multidisciplinary collaboration and supporting secure engineering processes.

Within digital engineering, this project applies model based systems engineering (MBSE), which is a set of standardized systems modeling techniques that can be used to support and develop the system requirements, design, analysis, and evaluation [18]. MBSE benefits the interdisciplinary design process by providing an authoritative source of truth and requirement-component traceability for segmented research and design [3, 15]. By focusing design on a standardized digital subsystem model, MBSE reduces complexity and dependency on Subject Matter Experts (SMEs) in the early stage development process [1, 16]. Digital models update automatically and work through a central platform, which helps to overcome the barrier of tool inter-operability and reduce the time and monetary costs of communication [3].

The connected traceability digital thread enabled by MBSE unifies the siloed data structures of traditional design, and supports complex integration of emergent system behaviors. Although these benefits are conceptual, previous case studies have found engineering benefits in the aerospace sphere. A study comparing document and model based systems engineering approaches for Submarine Warfare Tactical Systems (SWFTS) found that model-based digital engineering techniques posed statistically significant improvements on return on investment [2]. In particular, they saw 37% decrease in error detection during platform testing, indicating that the MBSE methods detected defects earlier in the design process than traditional ones. Also, an early digital thread implementation for a Boeing 777 saw reduced costs for design development after implementing MBSE [3]. In that case, the implementation resulted in 25% lower development time and 80% fewer assembly hours, while software costs were halved. Overall, these and other design processes conducted via MBSE benefit from traceable and automated change management which allows for earlier error detection, lower design costs, and faster product maturation [15].

This work leverages SysML to provide a cohesive description of the project throughout the product life-cycle development by standardizing the types and relations between system model artifacts [2]. SysML uses consistent sets of heuristics to define four basic pillars of a system's design: requirements, behavior, structure, and parameters. These different considerations are modeled using different diagrams and artifact types, depending on what information they provide about the final design and what logic is transferred between them. Within SysML, logic is defined as the flow of information or material between components which facilitate the physical system's actions [2, 6]. Defining, implementing, and tracking this logic appropriately is the primary challenge of MBSE which SysML helps to organize.

Given a modeling language, the MBSE procedure requires a framework for how to approach design development. The RFLP process is one such approach to build a cohesive system engineering model. The RFLP process starts with engineering a set of high-level requirements based on stakeholder needs to understand the full design space and its complexities, and provide a foundation for later traceability [19, 20]. This space forms the system *description*, and institutes the boundaries within which the design engineers can iterate. The requirements space is next developed with the resulting system behavior during functional analysis. Understanding of this behavior helps to establish the system activity, and evaluation of this activity validates that the system operations achieve the stakeholders' goals. Finally, the requirements and activities are analyzed to determine their contributions to the logical and physical design [6, 19]. This procedure ensures a direct, traceable connection between the stakeholder's original problem statement and the design solution.

The International Council on Systems Engineering (INCOSE), a leading professional body for systems engineers, has outlined Systems Engineering Vision 2035, which describes the current status of the systems engineering field and the goals which must be developed to meet the changing and increasingly global challenges faced [21, 22]. According to INCOSE, systems engineers must build standardized digital engineering practices which are repeatable, data-driven, and application-agnostic. These require common data storage and representation techniques, standardized artifact and analysis integrations, and architectures which facilitate automatic traceability that can be maintained throughout the product life-cycle [22]. This work contributes to these fundamentals by introducing a procedure for a conceptual-stage design process which is grounded in stakeholder demands and includes the background traceability artifacts necessary for system validation and verification. The approach engages SysML to imbue a standardized language and framework for managing the data underpinning design [1, 2]. The framework also formalizes a definition of artifact-to-artifact

relations, ensuring that the digital traceability between elements has a set meaning to facilitate communication-via-model. Employing an end-product-functionality focused approach allows the system requirement space to be built directly from the stakeholder needs and system tasks [23]. While MBSE case studies already exist for aircraft engine inlets, aviation electric fuel pumps, certification digitalization, and hydrogen aircraft fuel systems, this paper applies a traceability-first SysML/RFLP workflow to the IFS as a propulsion structural subsystem and structures the baseline model for later hybridization-oriented extension.

IV. Technical Approach

IFS modeling employed a two-stage approach, starting with more traditional spreadsheet based design processes which were then converted to digital graph-based design documentation in the MagicDraw environment using SysML. Beginning with a set of initial stakeholder needs, the RFLP method was applied to develop the full requirements space and system behavior so that the logical and physical architecture of the system could be modeled. Traceability was introduced between RFLP stages by establishing different relation types between model artifacts, such as requirements and activity blocks, to track the decision-making flow and dependencies through the complex system. The connective traceability was visualized through dependency matrices, which show the directional relationships between SysML components that govern validation and verification. The end result is an integrated digital system model, which can be used to quickly evaluate design choices and risks for more reliable collaboration and design development.

A. RFLP Approach

1. Requirements Engineering

The design procedure began with requirements engineering, where the system description was defined based on initial stakeholder needs. The goal of this stage was to develop a set of independent, hierarchical, and verifiable system requirements around these needs, and enable traceability between the original stakeholder's description and the final system definition. As this requirement space was developed, the system complexities are outlined in graphical form as visual diagrams to simplify multidisciplinary design.

The project started out with a list of sixteen stakeholder requirements based on SME knowledge of engine conceptual design from partners at Collins Aerospace. The first step was requirement decomposition, where the initial list was modified and expanded to fully define the system space outlined by the stakeholder's general specifications. Redundant items were condensed, while requirements that stipulated multiple design conditions were broken down into their component parts. Finally, any additional requirements implied by the original set were added to the system. These were referred to as 'derived requirements'. These additional requirements included those that would be necessary to set a weight budget for the conceptual design to provide a future comparison metric between the traditional and hybridized propulsion systems. An example decomposition is shown in Fig. 2, where the initial stakeholder request concerning fire management for the IFS was broken down into its composite conditions and regulations. The completed requirements space should address all business practices, system tasks, and failure modes discussed by the stakeholder, then consider any potential safety risks, and relevant performance metrics and components based on the desired system function and regulatory space. The procedural details are available in Table 2, which is a representative requirements normalization. The full table including all 16 stakeholder needs can be found in Appendix B. Upon completion, the full set of requirements were tabulated for further engineering and transferred to the solution domain. The stakeholder statements were preserved unchanged in the Problem Domain as the original source inputs. Requirement decomposition, ambiguity reduction, derivation of implied conditions, and assignment of requirement types were then performed in the Solution Domain, where the engineered UM requirements used for the remainder of the workflow were defined.

The second step of requirements engineering is to sort each solution domain requirement into different categories based on their design considerations. These categories for the baseline IFS model are listed in Table 3. Each categorized requirement was also assigned its type, describing what it dictated about the design space. Physical requirements were the simplest type, which described some attribute of the system such as the size or propulsion architecture. Functional requirements described some action completed by the system, such as 'contain fire' or 'attenuate noise pollution'. Performance requirements provided benchmarks for how well the system completed its function; for example the mass flow rate through a vent. Finally, design constraint requirements described any condition set for the process of designing or developing the product that must be satisfied during verification or certification. In contrast to the other, more formal requirement types which limited the product being designed, design constraints restrained the engineers doing the designing. These restrictions could include available materials, manufacturing processes, or expectations placed by

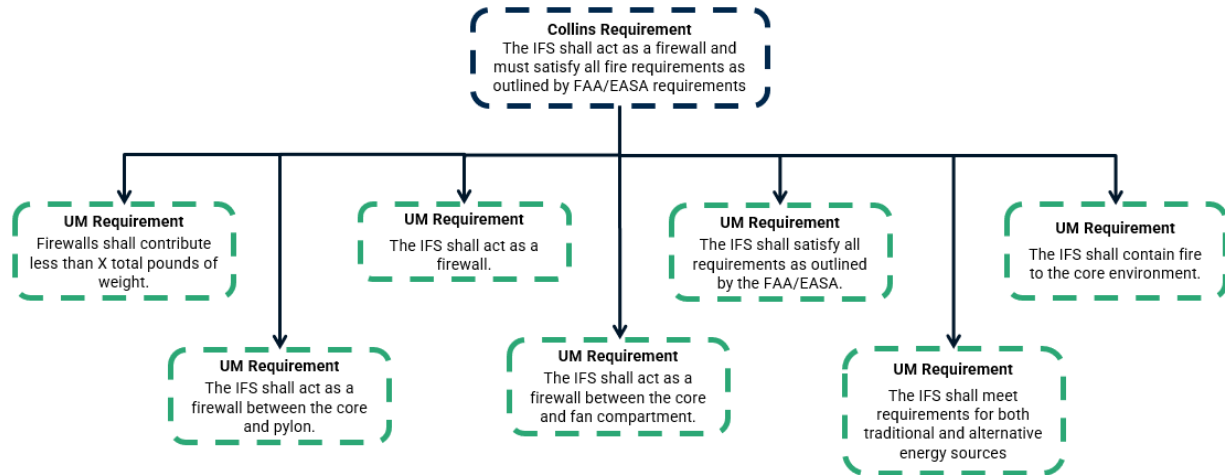


Fig. 2 Requirement decomposition for a stakeholder specification that the IFS act as a firewall. The initial requirement is broken down to identify which components the IFS must protect as a firewall, these are the implied requirements for the original stakeholder need.

company policies and the regulatory space. Sorting and allocating the requirements list by type in the above ways helped to organize the requirements space, validate that the desired stakeholder tasks were fully defined in the solution domain, and facilitated later tracking of where and how design changes cascaded through the system.

The next step of requirements engineering was to organize the solution domain requirement categories into hierarchies. To do this, the categories were each assigned a 'governing' requirement, which summarized the category's subsystem-level goal and role in the IFS' design space. As child requirements were derived from their parent requirement, high-level governing requirements were abstracted based on the category's solution domain contents *after* completing the initial requirement engineering. These governing requirements represented top-level engineered requirement descriptions rather than verifiable, detailed, clause-level requirement artifacts. As an example, the governing requirement for the regulatory requirements category was: '*R.1 The IFS shall have a clear path to certification*'.

After establishing each category's top-level requirements, the parent-child relations within each category were identified, and the child requirements were designated as sub-requirements of their parent. A requirement was determined to be a sub-requirement if it provided further specification (or *refinement*) for another requirement or was implied by another requirement. Child requirements were nested under their parent requirement during modeling, as will be described in Section IV.B. The formation of these hierarchies isolated and organized those requirements that would be the most tightly interrelated. Figure 3 shows the parent/child development, or local hierarchy, for the second maintainability requirement. Not all child elements in this local hierarchy are requirements of the same type. *M.1.1.1* is modeled as a physical requirement, whereas *M.1.1.2* and *M.1.1.3* are modeled as design constraints because they prescribe implementation choices or design-process limits used to satisfy *M.1.1*. These design constraint requirements are not solution-neutral functional requirements; they are representative of internal business practices and definitions, which enforce constraints on the design process as it was conceptualized for this baseline model. It should be noted that these design constraints are thus not universally necessary for an IFS model. More generally, parent-child structure in the local model captures refinement of the design space, not only requirement-to-requirement decomposition.

The final step of requirements engineering was to identify the relations between each requirement. Inter-connectivity was first established within the design categories by defining any relationships between requirements that had not already been established by the hierarchy's parent-child nesting. After this, the connections between requirements in different design categories were determined. Establishing this traceability enunciated couplings and tradeoffs within the system, and differentiating the types of connections helped to define the strength of those couplings.

Two types of relations were used to establish the traceability between requirements: derive relations and dependencies. The strongest connection type, derive relations, were applied when one requirement was implied by another that was not its parent. This often occurred when there were multiple sources for the same sub-requirement; only the tightest relation was identified in the hierarchy, and the remaining sources were connected via derive relations. The other type was called a dependency. This was the weakest relation, which was used when one requirement more loosely relied on

Table 2 Representative requirement-normalization table decomposing the stakeholder Problem Domain requirements to the Solution Domain UM requirements for the baseline IFS model. The full normalization table can be found in Appendix B.

Stakeholder Statement	Defect	Normalization Action	Solution Domain Requirement	Type
1. The air washed surface shall conform to the inner bypass duct loft lines within tolerances of +/- X in.	Ambiguous	Add	W.1.1 - The air washed surface shall contribute less than X to the GTOW.	Physical Requirement
	Non-Verifiable	Refine	S.1.1 - The limit load factor shall be at least X.	Performance Requirement
	Ambiguous	Derive	I.1.3 - The aerodynamic interface between the fan and IFS shall be maintained within +/-X tolerance.	Performance Requirement
			S.1.5 - Following engine failure, the IFS shall be able to support the windmilling loads for X.	Performance Requirement
			S.1.7 - The air washed surface shall conform to the inner bypass duct lines within X.	Performance Requirement
			S.1.8 - The air washed surface waviness shall be within X tolerance.	Performance Requirement
4. The IFS shall act as a firewall and must satisfy all fire requirements as outlined by FAA/EASA requirements	Ambiguous	Add	W.1.5.2 - Firewalls shall contribute less than X to the GTOW.	Physical Requirement
		Derive	M.1.1 - The IFS shall contain fire to the engine core environment.	Functional Requirement
	Compound	Split	I.1.1 - The IFS shall accommodate firewalls.	Functional Requirement
			I.1.1.1 - The IFS shall accommodate firewalls between the core and pylon.	Physical Requirement
			I.1.1.2 - The IFS shall accommodate firewalls between the core and the fan compartments.	Physical Requirement
			R.1.1 - The IFS design shall meet regulations for both traditional fuel and alternative energy systems.	Design Constraint
			R.1.2 - The IFS shall satisfy all fire requirements as outlined by the FAA/EASA.	Design Constraint

another for its implementation or designation [20, 24]. SysML relations always point from the dependent artifact to the source artifact, in this case from the dependent toward the dictating requirement. At the local level, requirement neighborhoods include both hierarchical parent-child structure and non-hierarchical cross-links such as dependency and derived-requirement relations; these are visualized in scoped group diagrams and in the dependency matrix presented in the Results section.

The result of requirements engineering was a fully defined Solution Domain that organized the minimum design requirements together with the governing design constraints needed to develop the most general baseline concept which

Table 3 Design categories for the IFS requirements model, and their description.

Design Category	Description
Interfaces	Requirements relating to the interface between components.
Latches	Requirements relating to the latches which provide access to the IFS.
Maintainability	Requirements relating to the lifespan, durability, and maintenance of the IFS.
Noise	Requirements relating to regulation of noise production.
Performance	Requirements relating to IFS performance and aerodynamics.
Regulatory	Requirements that must be met to establish a clear path to certification.
Structures	Requirements relating to configuration and loading.
Weight	Requirements relating to the weight budget for the IFS.

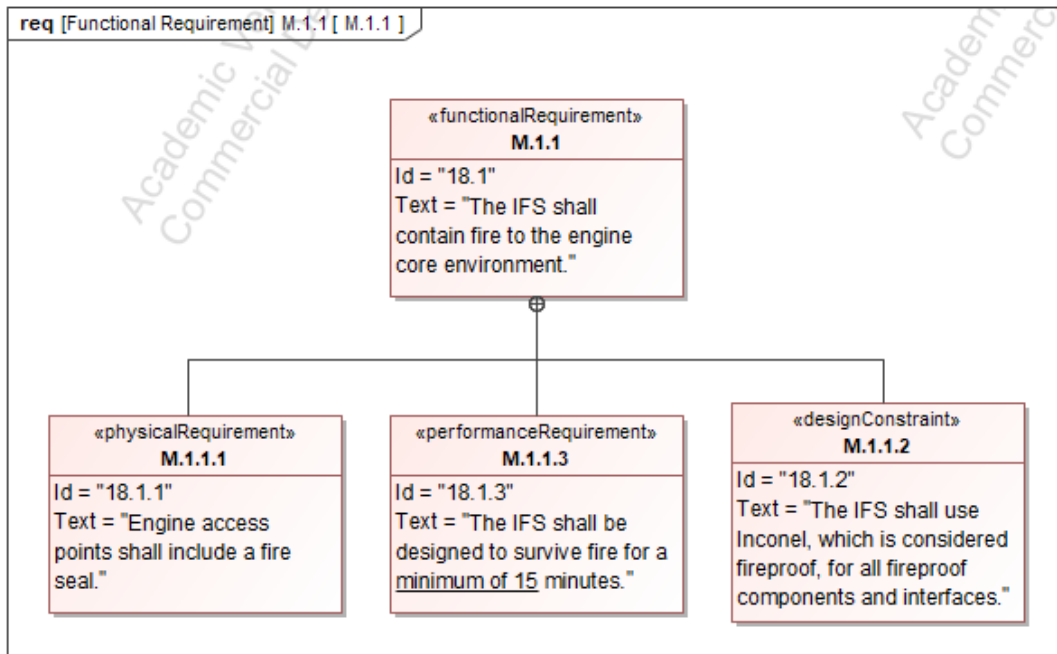


Fig. 3 Local decomposition around M.1.1, showing mixed child elements used to satisfy the parent statement. M.1.1.1 is a physical requirement, while M.1.1.2 and M.1.1.3 are design constraints that restrict how M.1.1 is implemented in the baseline concept.

satisfied the stakeholder needs. The engineered requirements correlated directly to the original stakeholder specifications and mapped the system inter-dependencies. After modeling, the requirements space provided a centralized source of truth which can be used to quickly run trade studies throughout later stages of the design process, and for validation and verification of the final design. During conceptual design, the final requirements list provided a basis for functional analysis.

2. Functional Analysis

Functional analysis is the process of defining the system behavior by describing the tasks which complete each requirement and decomposing them into discrete steps, called actions. The purpose was to define how the final design behavior serves the stakeholder needs, and ensure that the requirements space was satisfied with the necessary system activities. The requirements were first sorted into two categories in accordance with what they described about the system. Product requirements described system attributes and performance – i.e. what the system was and did. In parallel, process requirements defined how the system met its benchmarks [6]. The requirement '*contain fire to the engine core*' would be a product requirement, even though it describes an action. However, '*use Inconel for all fireproof*

parts' was a process requirement, because it directs the procedure for ensuring something is fireproof.

Next, each requirement was analyzed to determine its underlying intention. Based on these design goals, the requirements were assigned a basic function that needed to be met for their intention to be satisfied. These initial functions were very general and, in many cases, simply restated the requirement in the form of a task. The basic functions were then decomposed into discrete steps that would complete the desired system behavior. This process of refining the level of detail for the activity decomposition was repeated until a general physical system could be derived from the activity space. These decompositions could then be added to the digital system model to replicate the flow of actions conducted by the system. It is important to note that the final resolution of the activity breakdown should be set by the design engineer to match the scope of their analysis and the goals of the final digital model [19]. This is one of the benefits of the RFLP process: it is an agnostic method, and can be tuned to apply to most systems. What matters when using this process is that analysis of the functional models, and their traceability to the requirements, provide valuable insight into potential gaps and over-constraint in the design space for streamlined product development.

3. Logical and Physical Architecture

The ultimate step of the RFLP method was to construct the logical and physical architecture, which organized the flow of information and components that made up the system and its behavior. The physical architecture was made up of the general components needed to complete the activity breakdown. In contrast, the logical architecture was the connective flow of information, signals, and action that traversed the physical architecture. The goal was to build a digital map of the system that illustrated the design space visually, and traced the components of that space back to the functional and activity and requirement definitions which underlay them.

Systems engineering for the physical architecture proceeded directly from the functional analysis. The steps of each function decomposition were analyzed, and all components that would be needed to complete the tasks were listed. In addition to tangible design elements, the need for signals and messages was captured in the logical architecture, while the components that generate, transmit, or receive those signals were captured in the physical architecture or, when outside the IFS boundary, in the external interfacing systems.

For example, the FAA required certain alerts be enunciated to the pilot in case of fire in the engine fan compartment. The IFS thus required a sensor package to monitor temperature and pressure to complete this task, which then connected to components outside the IFS system boundary such as cockpit displays and a central computer to send signals. Although only the sensor package is part of the IFS system physical architecture, the external components were depicted in the physical architecture block diagram for completeness of the logical description and verification. However, an additional context model, in Fig. 10, identifies these as external interfacing components and not constituent IFS components. These interfacing components provide a more complete context for the logical architecture, in this case clarifying to where the sensor package transfers its damage signal. Including these components prevents the architecture from having any unterminated branches.

The architecture was fundamentally an interactive network flow chart visualizing the system and its function; where the physical architecture was represented by the chart's nodes, and the logical architecture by the directional branches connecting those nodes. Thus, the logical framework was developed from the list of physical components and functional analysis by assessing the nature of the interaction between connected stations. From the above emergency fire alert example, the physical components '*sensor package*' and '*central computer*' would be connected by the logical message '*record and communicate engine data*'. Where the physical architecture defined the process chains for the system's function, the logical construction imbued the architecture with the necessary actions and information to complete those chains.

Having defined the system components and logic, the architecture was ready for modeling. At this step the traditional engineering methods gain new value in the digital space and the benefits of the generic approach become apparent. By generalizing each feature, the architecture is able to represent all acceptable configurations within the design space rather than a single design selection. The implementation for this architecture is unusual in that it seeks to capture the structural loading cases, failure modes, and verifiable design constraints within the logical architecture. This ensures that loading, safety, and certification are reinforced as guiding parameters to the IFS design process. The modeling process also centralized the integrated design into a scalable, interactive environment that allows engineers to focus on their disciplinary concerns within the larger multidisciplinary system.

It should be noted at this point that although the systems engineering methodology has been laid out independently, digital modeling occurred simultaneously with the relevant RFLP step in parallel with the design process. The two stages of the technical approach were separated in writing only to ensure clarity of the methods used. In reality, each

engineering stage was ported to the digital environment as it was completed because the modeling process revealed additional details about the system as it was designed. The SysML tools and details of the digital modeling procedure are described next.

B. Digital Modeling

Once each RFLP stage had been documented, its features were added to the digital model to support design development. Model elements were developed and diagrammed for each step, and traceability was established between them. This supported iterative improvement and re-verification of the IFS design in the digital platform throughout systems engineering. Although there are numerous digital ecosystems for SysML modeling, this project uses *Dassault Systemes* Cameo MagicDraw because it offers versatile tools for traceability and subsystem integration [25]. The platform uses a set of guiding rules to dictate verification and validation strategies, which can be set and updated by the user. In combination with the ability to quickly generate diagrams across the system scale and perspective, this feature allows customizable, streamlined evaluations of the requirements space and final product.

1. Requirements Modeling

Both the initial stakeholder and the final system design requirements had to be modeled in the MagicDraw environment. The stakeholder requirements were modeled in the *Problem Domain*. These requirements were listed in an Excel spreadsheet, and then uploaded to MagicDraw to generate the requirement table. The table included the requirement description and its numbered label. Requirement blocks were automatically generated into the problem domain corresponding to each requirement in the table. The problem domain did not require any requirements engineering, sorting, or modeling. These requirements were included in the integrated systems model only to allow validation and verification of the final design architecture.

The same procedure was followed to model the *Solution Domain* requirement table for the baseline IFS design, after the requirements hierarchy was established as described in Section IV.A.1. Sub-requirements were nested under their parent in the table, which also automatically generated an *Owner/Owned* relation between the parent and child requirement blocks. These relations would later provide the structure for the local requirement hierarchies.

After completing the table, the requirement categories were sorted into different SysML packages which contained their constituent requirement elements and modeling diagrams. The requirement blocks were assigned their SysML type to match the type (physical, functional, etc.) set during the requirement engineering phase. The dependencies and derived relations identified for each requirement were next assigned within each block's specifications. At this point, the local requirements space was fully defined, and the model was made more efficient by adding constraint boxes to define any numeric values governing the system. This allowed values that recurred throughout the requirement space to be changed in one centralized location, which was then tracked through the applicable blocks. Constraints boxes were configured for numeric inputs, but could also be defined for time constraints or logical values to run different compliance checks.

The requirements for each category and any connected constraints were added to a local requirements diagram, which visualized the group hierarchy. These group diagrams contained the requisite intra-category relations for each set of requirements and could be scaled to focus only on their localized relations or expanded to show system-wide connectivity. As a final step, a full system requirements diagram was created that combined all local hierarchies into a larger model of the requirements space. Combining the local diagrams into a globally connected system allowed changes to one requirement or subsystem to be tracked through the larger IFS design, and facilitated more complete verification and traceability with the additional model artifacts.

2. Functional Modeling

The functional analysis for each requirement was modeled using activity diagrams, which applied the SysML standardized element types to map out each step of the system behavior. These diagrams answered the questions of 'how' and 'why' tasks should be completed during the design/build/maintenance cycle. Additional clarifying information was stored within the diagrams, such as the relevant *Solution Domain* requirements and governing regulations satisfied by the activity. Activity diagrams for product requirements often constrained design decisions, and therefore included iterative loops to identify, make, and verify those choices. In contrast, functional breakdowns for process requirements usually had straightforward start-to-end flow diagrams. This was because those activities generally restricted how the final product behaved rather than the design path taken to get there, and thus required only verification of the correct

final result. The activity diagrams were related back to the requirements they satisfied, so that design decisions could be correlated to system behavior and maintenance processes.

3. Architecture Modeling

The architecture was modeled using a SysML internal block diagram, where components of the physical architecture were defined as blocks and connected via flows, which sent logical signals between ports. Different ports were created for each block based on the logical connections they were assigned during the RFLP analysis. To do this, the architecture was first visualized using a basic flow chart, and then built in the MagicDraw system. After the basic architecture was complete, the components were organized into different subsystems. These helped to make the full architecture more legible, and made it easier to track signals through the system. The architecture was refined by grouping related components into their subsystems. This made the diagram easier to read and supported future design hybridization by defining the potential points of interaction with other propulsion subsystems. The organization will support modular comparisons between the traditional and hybridized IFS designs in future work.

A system boundary context diagram is a high level SysML representation which defines a system of interest (SoI) as a single entity and identifies all the external interfacing systems that interact with SoI and the exchanges that take place across its system boundary [20, 25]. This diagram is typically expressed using a block definition diagram or an internal block diagram in SysML, where the SoI is treated as the internal system or state, and all the interfacing elements are modeled as external blocks connected through item flows representing physical, informational or energy exchanges.

The system context diagram was initially built to explicitly define the system boundary of the IFS and resolve any ambiguity which might arise from the combined logical and physical architecture. While the architecture captures the system behavior, signal flows and interactions, it includes elements like Engine CPU and Cockpit Alert Displays which are necessary for the system operation but are not physically part of the IFS subsystem. This leads to misinterpretation of such elements as constituent components without a clear system boundary definition. Therefore, this context diagram addresses these issues by designating the IFS as the SoI and creating a clear system boundary from the external interfacing systems. This ensures that only the elements belonging to the IFS are treated as a part of its physical architecture. On the other hand, the external elements exchanging signals, loads and other modes of interactions are modeled outside the system boundary of the IFS.

The system boundary context diagram was built by first identifying the IFS as the SoI and extracting all the interacting elements and flows from the baseline logical and physical architecture of the IFS. All the elements were compiled and then evaluated based on whether it is physically part of the IFS or exists externally and interacts through an interface. The system elements which are not physically part of the IFS but required for system operation were classified as external interfacing systems and the interacting logic was represented using item flows corresponding to structural loads, control signals, sensor data, interfaces and hazard interactions.

C. Traceability

The goals of traceability were multifaceted, connecting through and between levels of analysis. The system requirements space had to be traced back to their originating stakeholder requirements, while the inter-requirement connections were also traced to track couplings within the requirements space. The system activity diagrams were also traced back to their requirements, to verify that the requirements space was fully satisfied. Finally, the architecture components had to be related to their founding activity diagrams to ensure that the system behavior could be completed. Traceability was therefore established through different relations and diagrams within the MagicDraw environment, which will be discussed next.

1. Requirement to Requirement Traceability

Requirements in the *Solution Domain* were traced back to the *Problem Domain* using explicit cross-domain traceability links in Cameo. In this project, those links were implemented as *Allocate* relations to preserve a clear mapping from the original stakeholder requirements to the engineered *Solution Domain* requirements while keeping the two domains distinct. This use of *Allocate* is a project-level traceability convention for cross-domain mapping, not the parent-child requirement hierarchy used within the *Solution Domain*. Traceability between the *Problem Domain* and *Solution Domain* requirements space was visualized using an allocate matrix and a requirement flow diagram. The requirements traceability matrix, Fig. 4 shows direct links between stakeholder requirements, in rows, and system requirements, in columns. Arrows positioned in the matrix indicate the presence and direction of the traced relationship.

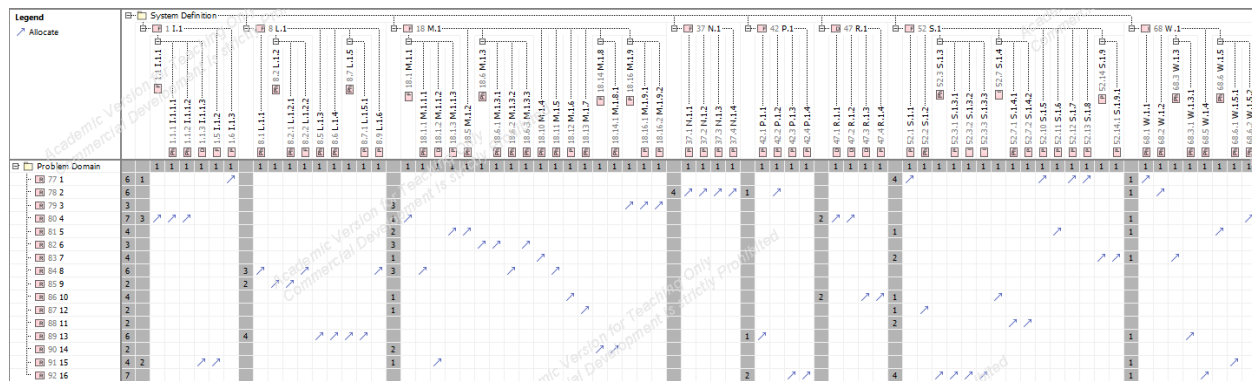


Fig. 4 Stakeholder requirements traceability matrix showing the cross-domain traceability links between the Problem Domain and Solution Domain requirements in MagicDraw.

Traceability was also established within the *Solution Domain* requirements space. As previously discussed, owned, dependency and derived requirement relations were used to establish different relation strengths within the system, and visualized in the requirement diagram. A traceability matrix and requirement flow diagram were also produced for the requirement space. The matrix showed all relations between requirements within the system. This allowed patterns and points of strong coupling between sub-disciplines to be identified within the system. However, the flow diagram only shows the containment hierarchy within the Solution Domain so that it provided a complementary view of the full requirement system structure without the added complexity of the cross-domain linkages.

2. Requirement to Function Traceability

Traceability was established between the activity diagrams and their originating requirements. Activity diagrams for iterative processes which represented design choices for a product requirement, were connected to their source requirements using *Satisfy* relations. These relations define a flow of information where the dependent diagram completes some task which realizes the source requirement [24]. *Satisfy* relations were also used to connect decision blocks to any requirement that their check supported. This strengthened requirement verification, and further integrated system complexity into the digital environment.

Activity diagrams for processes which did not check design decisions, particularly for process requirements, were connected using *Verify* and *Trace* relations. *Verify* relations were used when the activity diagram provided clarification or procedural information that must be verified for maintenance or design and build processes[24]. *Trace* relations defined general dependencies between model artifacts, and were applied when the activity diagrams outlined functional or performance characteristics for the final product. These relations thus indicated what the diagrams could tell users about the system, and provided the background structure for requirements validation at throughout design progression. By maintaining the model iteratively throughout this process, the traceability could provide validation at multiple levels of fidelity.

3. Function to Architecture Traceability

The architecture components were traced to the activity diagrams and component blocks that required them to validate system behavior. SysML diagram blocks were related back to the activity using *Allocate* relations, which defines the source activity diagram as responsible for the dependent architecture block [20]. These relations were general, so that connections could be identified between any two activity and structural components, for a thorough understanding of how the design model conducts the intended behavior. Once established, these could be used to represent the design process for low-cost digital evaluations of potential design selections or configurations. This process was extended to investigate an option for hybridization of the baseline IFS in a second phase model.

D. Phase 2: Hybridization Scoping

The project was structured in two phases: first building the baseline IFS model presented in this paper, and then defining how that model would be extended to represent the design process for a parallel HEP architecture in subsequent

work. The RFLP process lent itself to an iterative design approach, as models could be updated to add resolution or introduce new variations. This preserved all of the traceability and complexity derived from the first phase and provided insight into points in the architecture and requirements space where significant changes could be expected. Rather than starting from only the initial stakeholder requirements, the second phase used the baseline model as a jumping-off point to facilitate faster-paced design development of a novel configuration.

The selected hybridized architecture is a mechanically integrated parallel hybrid turbofan engine, where the accessory gearbox connecting the electrified and turbo-shaft drive-trains is mounted outside the fan case. Although conceptual level design usually proceeds from analysis of historical aircraft data [26], novel propulsion system design is hampered by the relative dearth of specifications and flight data for hybridized aircraft. To overcome this issue, the designers turned to concepts which used similar technologies at less demanding power draws. Evaluation of MEA technologies and electric motor designs and airframes illustrated the expected changes to fire zones, maintenance, safety, and power management that would impact the requirements space [4, 5]. In particular, these highlighted the new role played by the temperature vs space tradeoff which dictates power component design. As power needs increase and component sizes grow, the industry has turned to smaller and more modular designs [10, 27]. Modularity gives engineers more control over the placement, and thus ambient conditions of electric power components. These insights were used to define the hybrid electric propulsion requirements for the Phase 2 IFS design model.

V. Results

The results presented in this section correspond to the completed baseline IFS model; discussion of hybridization is limited to how the baseline model was structured to anticipate future change impacts and extension needs.

The first model output, following requirements engineering, is the *Solution Domain* requirements space demonstrated with the requirements diagram. The full requirements diagram for the baseline IFS design is shown in Fig. 5. The different color-coordinated blocks represent the different requirement groups: dark blue for maintainability, yellow for interface, orange for regulatory, purple for noise, light green for latching, pink for structural, dark green for performance, and light blue for weight requirements. The connections between groups may be isolated to make the relations easier to track, but the size of the requirements space means that this diagram's primary use is to illustrate the scale and complexity of the system design problem. At completion, the IFS *Solution Domain* contains 76 requirements and 212 relations, which present a significant challenge to trace and manage in a single diagram or analysis.

The burdensome size of this requirements space is precisely why digital modeling lends value to this design process; the requirement diagrams can be swiftly generated at the individual and category scales, and tuned to the scope of interest. This provides engineers flexibility to consider the requirement space at any resolution or perspective. The background connectivity established with the relations and block packages at this stage allows the digital environment to manage out-of-scope system complexities throughout the continuing design process, while disciplinary engineers focus only on their specialized analyses. The resultant centralization of the design definition in a scalable format improved the organization and traceability of the conceptual baseline IFS design process.

Figure 6 depicts an example local requirements diagram for the interface category. The diagram includes additional external requirements which have relations with the interface requirements and influential constraints. This view prevents design decisions relating to interfacing for the IFS from omitting important external factors. The limited scope also focuses early evaluation of the local hierarchies for completeness and accuracy. If an engineer wanted a more modular view, they could restrict the diagram to only show contained requirements and constraint values, or omit nested requirement levels. Updating the view and scope of the model does not change the central definition, or affect the views of other engineers working with the system. In this way, the model helps to prevent common pitfalls of traditional document-based design.

The activity diagrams display the flow of actions for each system task. Figure 7 is a diagram for the activity associated with product requirement *M.1.1.1*. The diagram displays the requirement to be satisfied and the relevant materials, regulations, and tests which support the activity as text boxes as supportive design documentation to explain the activity flow. The associated activity is to *Ensure engine access points contain seals* and involves the following actions: identify points which require seals, check that Inconel is used for fire rated seals, verify that all engine access points contain seal, and evaluate seal effectiveness. This activity iterates over design selection, validation, and testing so that the only way to logically conclude the process is to satisfy the intention of the original requirement. The intention is not for this particular flow to be conducted by any one engineer or designer, but to follow the IFS throughout the design life-cycle. By completing each of these steps throughout the design/build/test process, the final product will be validated for requirement *M.1.1.1*.

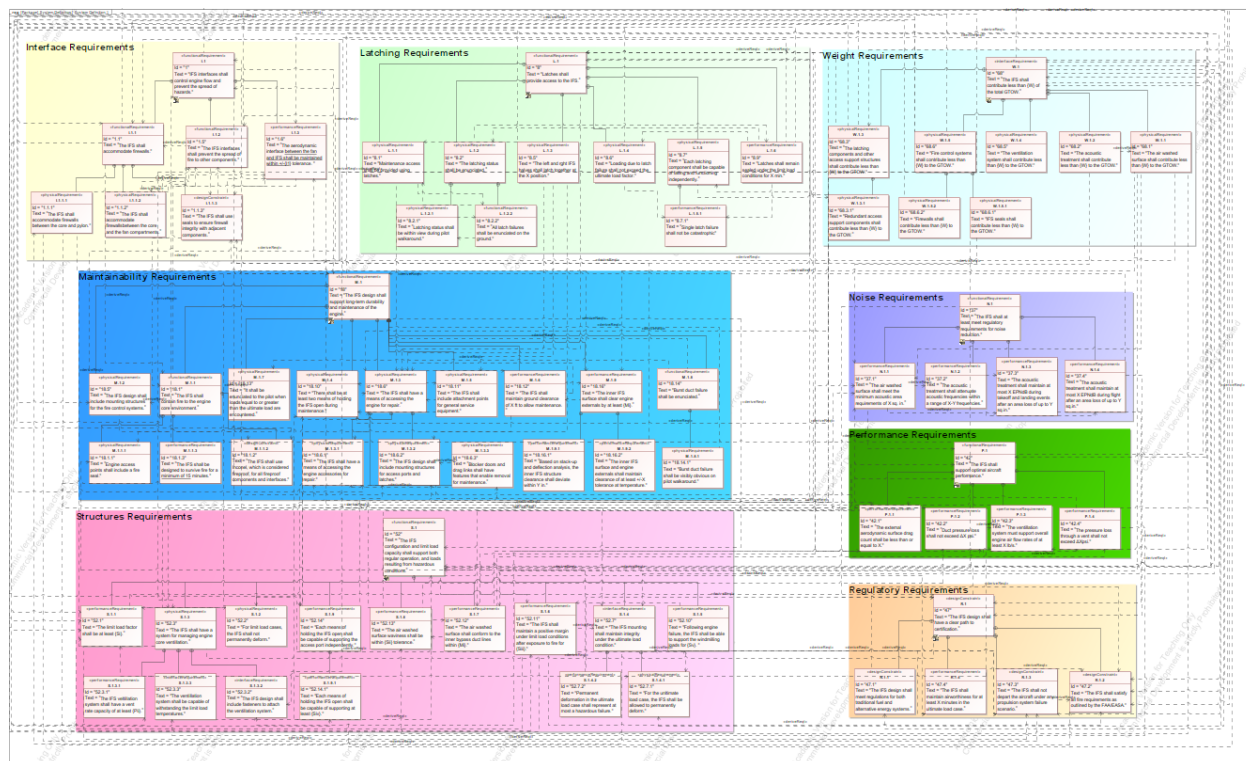


Fig. 5 Full requirements diagram for the baseline IFS design, showing the different design categories and the traced connections.

Similarly, Fig. 8 is an activity flow associated with the process requirement *M.1.1.2*. The activity is to *use Inconel for all fireproof components and interfaces* which satisfies a design constraint imposed by business procedures and the regulatory space. This condition must be satisfied for the design to be certifiable and thus considered valid, so it is included in the *Solution Domain* requirements and activities. The actions included in this activity are: specify which materials must be fireproof, specify Inconel as their required material, fit identified pieces with Inconel, extract any necessary structural information for testing. This activity is also intended to follow the IFS through the design life-cycle, but the requirement can be satisfied earlier in this cycle than the product requirement because it does not imply any testing conditions. Analysis of these differences between activities can thus reveal additional details about requirements and how they are satisfied in the design process.

Figure 9 is the internal block diagram representing the IFS physical architecture, logic, and interfacing with surrounding systems. It collectively showcases the system behavior by organizing the system and surrounding components and the associated flow of information and signals. The primary subsystems represented in the baseline IFS model include the Electronic & Monitoring System, Access & Latching Equipment, Protection Subsystem and the Engine Compartment which includes the Structural Subsystem, representative IFS block and the Fan Subsystem. The fan subsystem, pilot, engine CPU, cockpit alert display, and visual inspection are external nodes which interface with the logical flow of the IFS and are included for validation and completeness. Component definitions are generalized and maintained at an abstraction level such that the model is ensured to be complete, comprehensive and flexible. At the same time, any potential interfaces and sub-systems that could be identified were included to facilitate later decision making. By calling out the interfaces between components and signals, the model illustrates how component changes implicate changes to the surrounding system integration. This helps prevent interfaces from being overlooked during rapid development points, and reduces design risk.

The logical architecture represents the functional connectivity and information flow that propagates through the physical architecture. These signal flows include the transmission of electrical power, control commands and monitoring signals enabling coordinated system operation. For example, in the event of a fire, the fire detection signal sends a signal to the fire containment and extinguishing systems. Concurrently, status and alert signals are sent through the CPU and enunciated to the pilot via the cockpit display. This ensures that all regulations for fire management and requirements

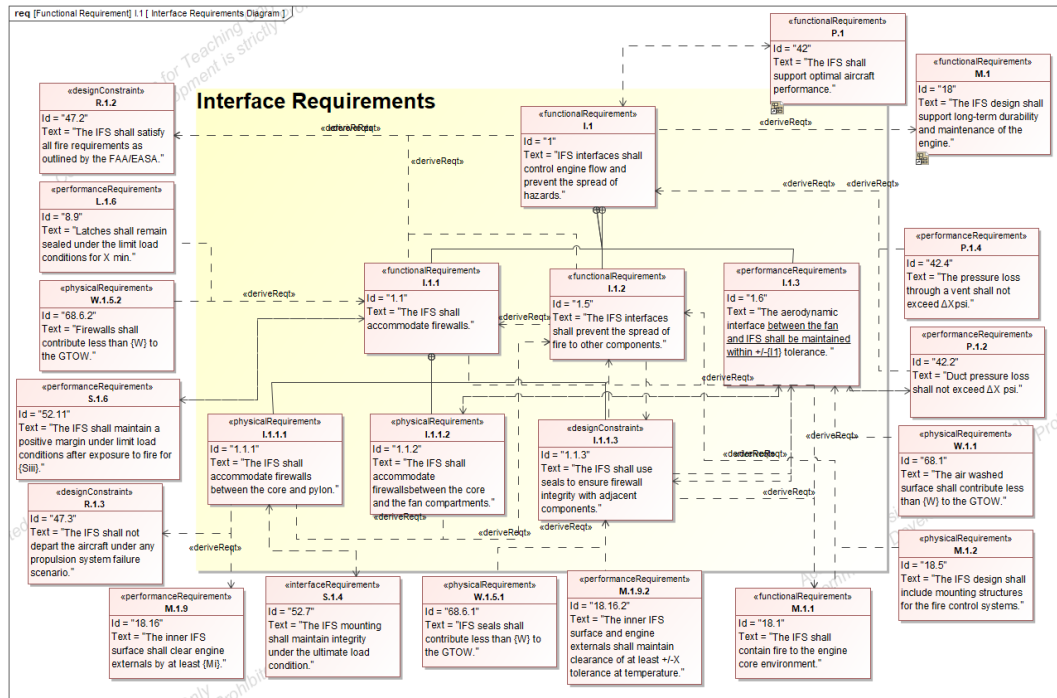


Fig. 6 Example local requirements diagram for the Interface category. The model includes the group's requirement blocks, intra-category connections between requirements, and the influential constraints for the contained requirements.

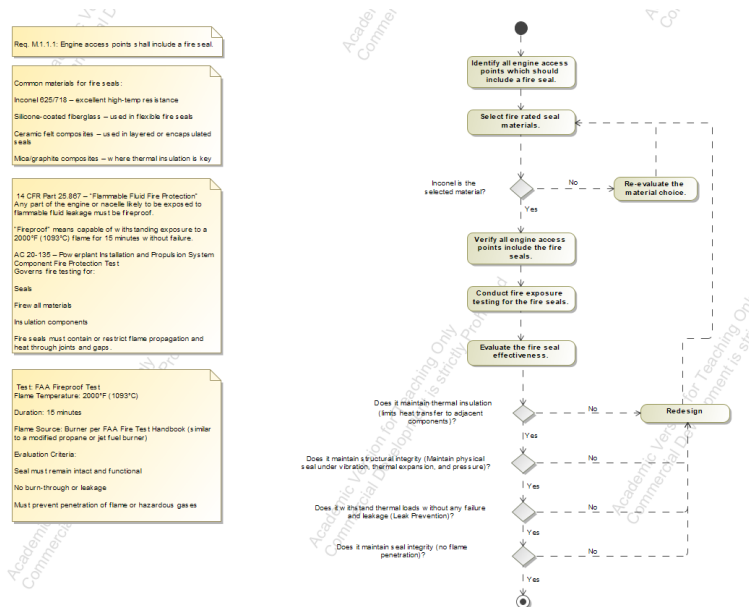


Fig. 7 Activity diagram for requirement *M.1.1.1*, a product requirement for the baseline IFS model.

for failure mode enunciation are carried out, and the stakeholder needs are thus validated. All together, these logical flows traverse the physical architecture to realize the system behavior and maintain traceability to the corresponding requirements of the IFS.

The resulting system boundary context diagram shown in Figure 10 establishes a formally defined SoI by distinguishing the IFS as an internal system and explicitly modeling all the boundary crossing interactions with the

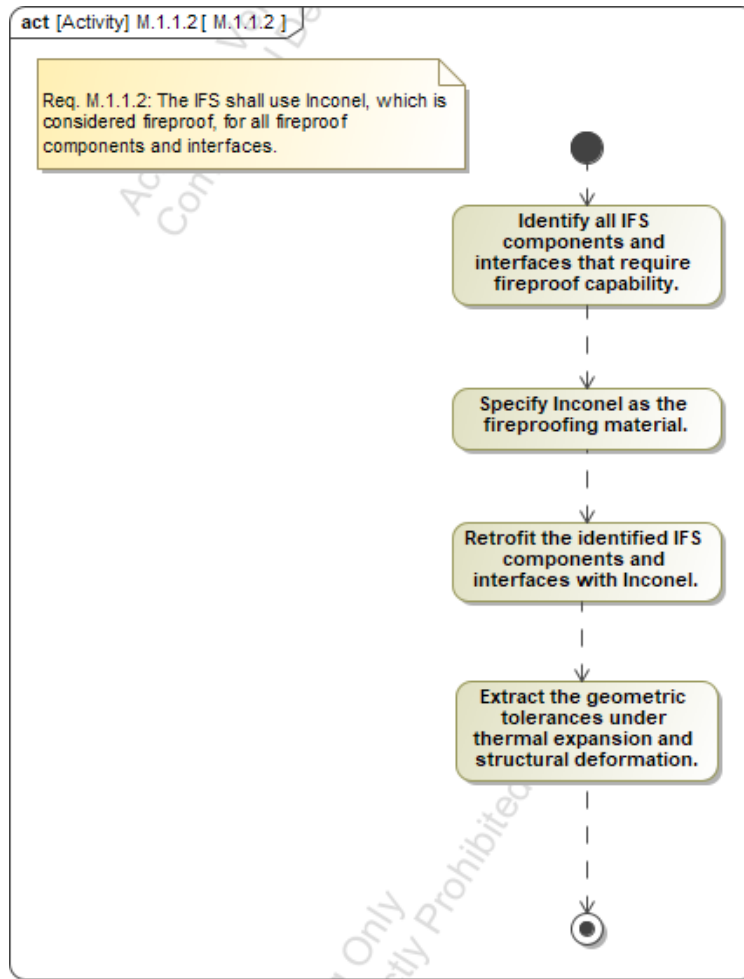


Fig. 8 Activity diagram for requirement *M.1.1.2*, a process requirement for the baseline IFS model.

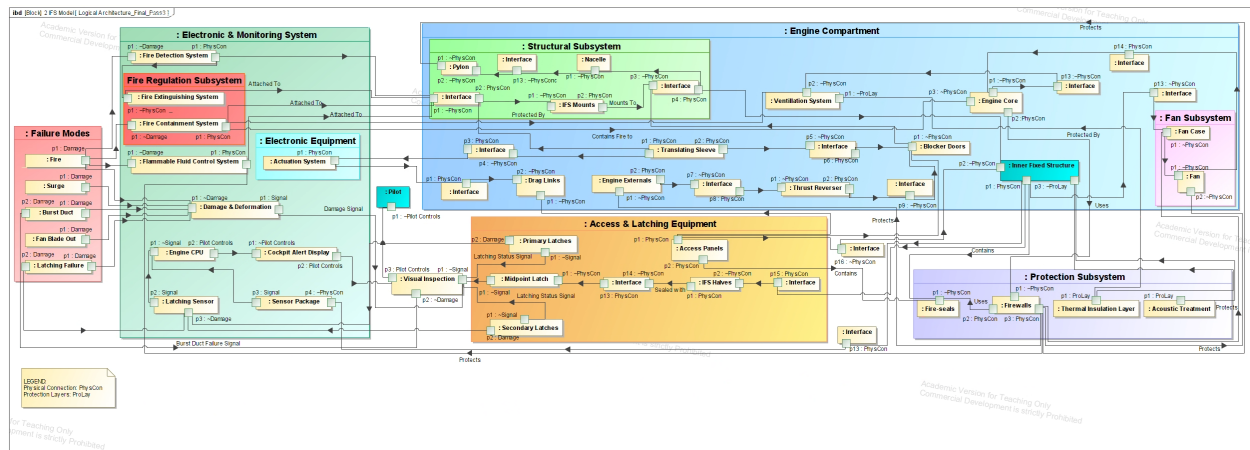


Fig. 9 SysML diagram of the full logical and physical architecture for the baseline IFS model. The physical components are represented as blocks, and the logical architecture make up the flows between those blocks.

external interfacing systems. The external systems are abstracted into higher level blocks including Airframe/Nacelle Interface, Engine Core, Ventilation System, Fan Subsystem, Electronic & Monitoring System, Hazard Sources and

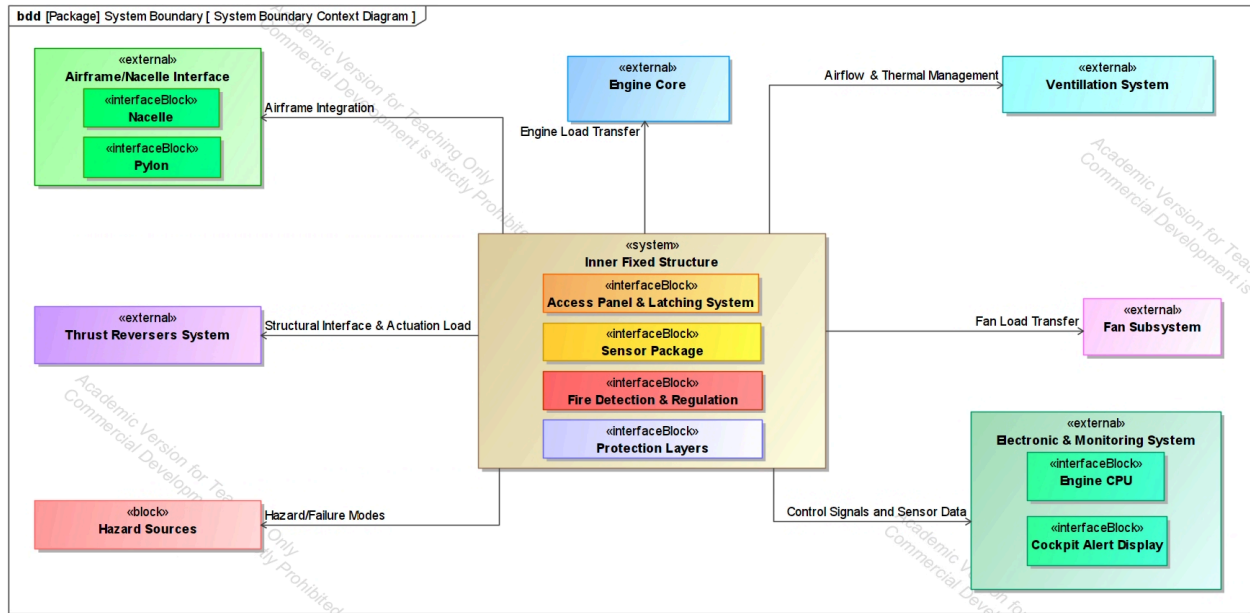


Fig. 10 SysML System boundary context diagram for the baseline IFS model. The IFS is represented as the system of interest, and external interfacing systems define the flows of signals, loads, and interactions across the system boundary.

Thrust Reverser System. The internal elements of the IFS include the Access Panel & Latching System, Sensor Package, Fire Detection & Regulation and Protection Layers. The interactions between the IFS and the external systems are represented by item flows which capture distinct exchange types including structural load transfer and interfaces, sensor data, control signals, aerodynamic and thermal interactions. The hazard sources shown in Figure 10 are modeled to represent the failure inducing interactions including burst duct, fire, fan blade out, surge, latching, damage and deformation.

This SysML diagram showcases and enforces a strict separation between the system composition and system interaction. This ensures that only the physically contained elements constitute the IFS and the external elements which exchange flows are modeled as interfacing systems. The abstraction and grouping of external systems reduce the complexity of the diagram while preserving interaction fidelity. This enables a consistent, straightforward and clear interpretation of the system boundaries and interaction flows. This results in improved architectural coherence, clearer traceability between the corresponding requirements and interfaces and a well defined foundation for subsequent logical and physical architecture development for complex aircraft components and systems.

The inter-requirement traceability is illustrated by the requirement map and the dependency matrix. The requirement map, shown in Fig. 11 depicts the full requirements space hierarchy stemming from the *Solution Domain* requirements package. The flow represents the ownership relations in the model. This diagram is a useful way to evaluate where requirements are concentrated in the design space.

The dependency matrix shown in Fig. 14 identifies the dependency and derive requirement relations between the interface and latching categories on the rows and the maintainability category on the columns. This version of the diagram has been reduced from the full form to make it more legible, and to isolate a set of highly interconnected systems. Because the latching and interfacing requirements dictate how the IFS is accessed and sealed, including during maintenance access, these categories experience a lot of coupling. The customizable nature of the MagicDraw environment allows this and other comparisons to be made directly from the same diagram by expanding different sections and updating the scope of the rows, columns, or elements. These scalable and modular aspects of the digital environment thus help to visualize and evaluate the system connectivity and evolution.

The activity diagrams were connected back to their originating *Solution Domain* requirements via satisfy constraints. The traceability between these components can be visualized in the satisfy matrix. An example of this matrix is Fig. 13 for the interface category requirements and activities. As this diagram shows, multiple requirements can be satisfied by the same activity. Additional diagrams can be produced for more granular views, to show relationships between

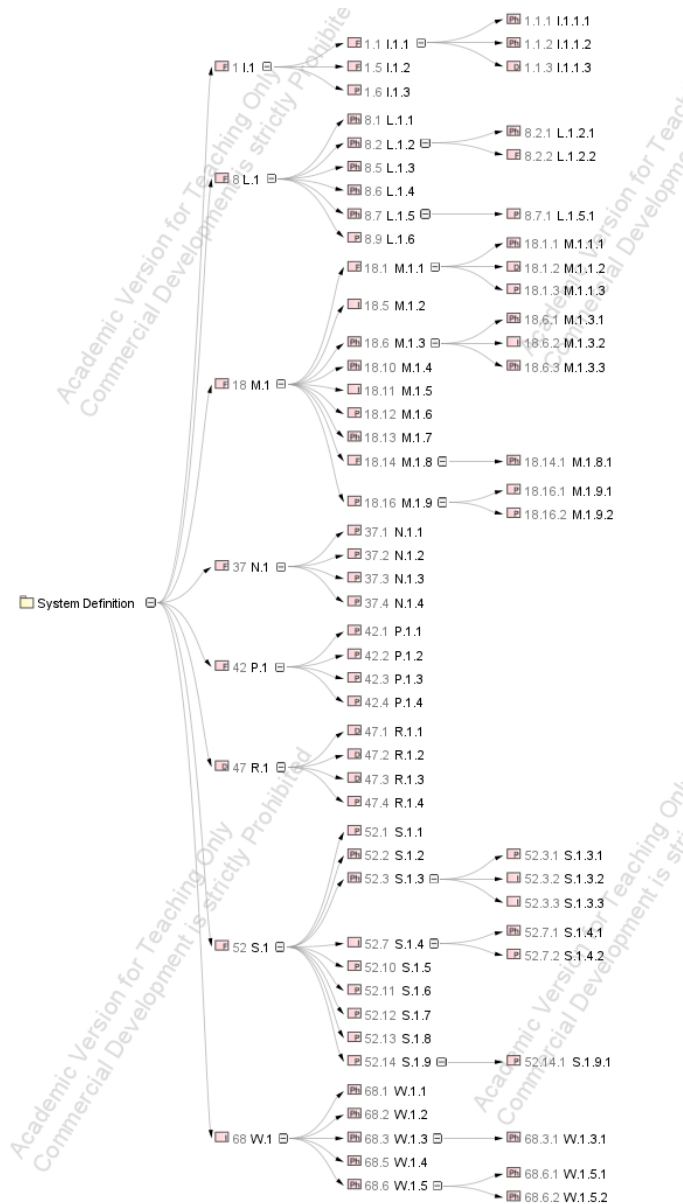


Fig. 11 Requirement map showing the flow of ownership representing the full requirement hierarchy for the *Solution Domain* requirements space.

individual actions and flows and their original requirements. The satisfy matrix is a useful tool for quickly verifying that the requirement space is fully supported by the system behavior throughout its life-cycle. When changes to the activity or requirement definitions leave some requirement un-validated, the satisfy matrix can help to identify where additional updates can rectify the inconsistency. As such, the layered connectivity established supports change management and traceability of conceptual propulsion designs.

In addition to preserving traceability between model layers and internal analyses, the digital model's scalability and modularity permit low-risk project collaboration. Design models representing proprietary or unnecessary design information can be reduced prior to distribution for outside partners. This enables more control over what and how information is shared during collaboration, eases the process of re-implementing artifacts from back-up or associated models if data is lost, and facilitates concrete representations of design iterations in place of documentation-based change logs. This use case was observed during project execution: individual engineers responsible for different RFLP stages were able to work with reduced versions of the model and then combine them into a single file for final traceability.

VI. Conclusions

This paper demonstrated a structured workflow that successfully applied the RFLP process to develop a baseline, hybridization-aware integrated systems model of the IFS, with explicit end-to-end traceability across requirements, behavior, and architecture artifacts in Cameo MagicDraw. The final digital platform stems from defined stakeholder needs, and serves as an integrated design document that combines the design considerations and requirement engineering across engineering disciplines. The key benefits of this digital engineering approach for conceptual phase design can be understood in terms of the model's versatility and generality. The block diagram is an abstract reference architecture for the class of configurations considered in this study. It captures common structure, interfaces, and constraints, while configuration-specific variants are introduced through controlled model updates. The model also streamlines design validation by providing an authoritative source of truth between iterations. This reduces the communication burden between SMEs in different fields of engineering during the early design phase when numerous interconnected changes are considered. Each agent can implement and evaluate their potential choices independently within the digital environment, facilitating collaboration with fewer delays. The models are scalable; their scope can be restricted by a user to focus on their piece of the system while the model handles the complexities of other disciplines. This simplifies the experience for each individual contributor to a project, while ensuring that inter-dependencies, verification, and validation are managed systematically.

This work contributes to a growing literature on practice for applying MBSE to propulsion system design by establishing a workflow structure, but much continuing work is needed. In the next phase of work, this framework will be applied to develop a second model for an IFS associated with a parallel HEP configuration, enabling direct evaluation of hybridization-driven requirement, interface, and architecture changes. However, additional research is needed to determine how to manage the unique challenges of hybridized design more generally. As definition of the initial requirements for the hybridized phase progressed, it became clear that these more integrated designs require more initial knowledge of the propulsion concept to begin requirement engineering. The additional components and tradeoffs implicated by high-power electronics introduce greater uncertainties, which must be tracked through both the system of interest and the design procedure. Future work will investigate appropriate methods for identifying, representing, and managing these uncertainty sources for early-stage conceptual design.

Another key lesson learned during this project was that the problem of scope develops very quickly in the requirements space, even for a non-moving structural propulsion component like the IFS. Further investigation is needed to identify the appropriate guardrails for defining a system model's scope and interfacing definitions. These would help to define metrics for model sufficiency, and support the implementation of formal reuse practices [22]. All of these research outcomes would serve to advance INCOSE's Vision 2035 by meeting systems engineering challenges for establishing integrated digital systems engineering as standard practice. This work contributes a repeatable, hybridization-aware RFLP-based MBSE workflow for propulsion subsystem design, enabling traceability-driven system modeling from stakeholder requirements to component-level architecture for complex aircraft propulsion systems.

Appendix A. Stakeholder/Problem Domain Requirements Table

Table 4 Table of stakeholder requirements. These text requirements made up the problem domain and served as the original source of traceability for validation.

Item	Stakeholder Text
1	The air washed surface shall conform to the inner bypass duct loft lines within tolerances of +/- X in
2	The air washed surface shall meet the minimum acoustic area requirement of X in ²
3	The Inner IFS surface shall clear engine externals by at least X in (with deviations down to Y in based on stack-up and deflection analysis)
4	The IFS shall act as a firewall and must satisfy all fire requirements as outlined by FAA/EASA requirements
5	The IFS shall be capable of being exposed to the effects of fire and must withstand those effects under load for at least X min
6	The IFS shall have a means of opening and closing such that the engine and engine accessories may be accessed for maintenance and repair
7	The IFS shall have two means of holding the IFS open during maintenance activities
8	The IFS shall use latches to provide maintenance access
9	Latches shall be configured such that unlatched latches are visibly obvious during a pilot walk-around
10	The IFS shall not depart the aircraft under any propulsion system failure scenario (FBO, burst duct. Fire, etc.)
11	For limit load cases, the IFS shall not permanently deform
12	For ultimate load cases, it shall be allowable that the IFS permanently deform
13	The IFS shall be capable of performing its intended functions with one, latch, hinge, or bumper failed
14	The IFS shall have a mechanism or means of visible indication of a burst duct failure that is visible during a pilot walk-around
15	Seals shall be provided to the IFS design to ensure firewall integrity with adjacent structural components
16	The IFS shall have ventilation features as necessary to meet the ventilation requirements of X lb/s

Appendix B. Requirement Normalization Table

Stakeholder Statement	Defect	Norm. Action	Solution Domain Requirement	Type
1. The air washed surface shall conform to the inner bypass duct loft lines within tolerances of +/- X in.	Ambiguous	Add	W.1.1 - The air washed surface shall contribute less than X to the GTOW.	Physical Requirement
	Non-Verifiable	Refine	S.1.1 - The limit load factor shall be at least X.	Performance Requirement
	Ambiguous	Derive	I.1.3 - The aerodynamic interface between the fan and IFS shall be maintained within +/-X tolerance.	Performance Requirement

			S.1.5 - Following engine failure, the IFS shall be able to support the windmilling loads for X.	Performance Requirement
			S.1.7 - The air washed surface shall conform to the inner bypass duct lines within X.	Performance Requirement
			S.1.8 - The air washed surface waviness shall be within X tolerance.	Performance Requirement
2. The air washed surface shall meet the minimum acoustic area requirement of X sq. in.	Ambiguous	Add	W.1.2 - The acoustic treatment shall contribute less than X to the GTOW.	Physical Requirement
	Ambiguous	Derive	N.1.1 - The air washed surface shall meet the minimum acoustic area requirements of X sq. in.	Functional Requirement
			N.1.2 - The acoustic treatment shall attenuate acoustic frequencies within a range of X-Y frequencies.	Performance Requirement
			N.1.3 - The acoustic treatment shall maintain at most X EPNdB during takeoff and landing events after an area loss of up to Y sq.in.	Performance Requirement
			N.1.4 - The acoustic treatment shall maintain at most X EPNdB during flight after an area loss of up to Y sq.in.	Performance Requirement
	Non-Verifiable	Refine	P.1.2 - Duct pressure loss shall not exceed ΔX psi.	Performance Requirement
3. The Inner IFS surface shall clear engine externals by at least X in (with deviations down to Y in based on stack-up and deflection analysis)	Compound	Split	M.1.9 - The inner IFS surface shall clear engine externals by at least X.	Performance Requirement
			M.1.9.1 - Based on stack-up and deflection analysis, the inner IFS structure clearance shall deviate within Y in.	Performance Requirement
		Derive	M.1.9.2 - The inner IFS surface and engine externals shall maintain clearance of at least +/-X tolerance at temperature.	Performance Requirement
4. The IFS shall act as a firewall and must satisfy all fire requirements as outlined by FAA/EASA requirements	Ambiguous	Add	W.1.5.2 - Firewalls shall contribute less than X to the GTOW.	Physical Requirement
		Derive	M.1.1 - The IFS shall contain fire to the engine core environment.	Functional Requirement
	Compound	Split	I.1.1 - The IFS shall accommodate firewalls.	Functional Requirement
			I.1.1.1 - The IFS shall accommodate firewalls between the core and pylon.	Physical Requirement
			I.1.1.2 - The IFS shall accommodate firewalls between the core and the fan compartments.	Physical Requirement
			R.1.1 - The IFS design shall meet regulations for both traditional fuel and alternative energy systems.	Design Constraint
			R.1.2 - The IFS shall satisfy all fire requirements as outlined by the FAA/EASA.	Design Constraint

5. The IFS shall be capable of being exposed to the effects of fire and must withstand those effects under load for at least X min.	Ambiguous	Add	W.1.5 - Fire control systems shall contribute less than X to the GTOW.	Physical Requirement
	Compound	Split	S.1.6 - The IFS shall maintain a positive margin under limit load conditions after exposure to fire for X.	Performance Requirement
			M.1.1.3 - The IFS shall be designed to survive fire for a minimum of 15 minutes.	Performance Requirement
	Ambiguous	Derive	M.1.2 - The IFS design shall include mounting structures for the fire control systems.	Physical Requirement
6. The IFS shall have a means of opening and closing such that the engine and engine accessories may be accessed for maintenance and repair.	Compound	Split	M.1.3 - The IFS shall have a means of accessing the engine for repair. M.1.3.1 - The IFS shall have a means of accessing the engine accessories for repair.	Physical Requirement Physical Requirement
		Derive	M.1.3.3 - Blocker doors and drag links shall have features that enable removal for maintenance.	Physical Requirement
7. The IFS shall have two means of holding the IFS open during maintenance activities.	Ambiguous	Add	W.1.3 - The latching components and other access support structures shall contribute less than X to the GTOW.	Physical Requirement
		Derive	S.1.9 - Each means of holding the IFS open shall be capable of supporting the access port independently. S.1.9.1 - Each means of holding the IFS open shall be capable of supporting at least X.	Performance Requirement Performance Requirement
	—	—	M.1.4 - There shall be at least two means of holding the IFS open during maintenance.	Physical Requirement
8. The IFS shall use latches to provide maintenance access	Ambiguous	Add	M.1.1.1 - Engine access points shall include a fire seal. M.1.3.2 - The IFS design shall include mounting structures for access ports and latches. M.1.5 - The IFS shall include attachment points for general service equipment.	Physical Requirement Physical Requirement Physical Requirement
	—	—	L.1.1 - Maintenance access shall be provided using latches.	Physical Requirement
	Ambiguous	Derive	L.1.2.2 - All latch failures shall be enunciated on the ground.	Physical Requirement
			L.1.6 - Latches shall remain sealed under the limit load conditions for X min.	Performance Requirement
9. Latches shall be configured such that unlatched latches are visibly obvious during a pilot walk-around.	Compound	Split	L.1.2 - The latching status shall be enunciated.	Physical Requirement
			L.1.2.1 - Latching status shall be within view during pilot walkaround.	Physical Requirement

10. The IFS shall not depart the aircraft under any propulsion system failure scenario (FBO, burst duct. Fire, etc.)	—	—	R.1.3 - The IFS shall not depart the aircraft under any propulsion system failure scenario.	Design Constraint
	Compound	Split	S.1.4 - The IFS mounting shall maintain integrity under the ultimate load condition. R.1.4 - The IFS shall maintain airworthiness for at least X minutes in the ultimate load case.	Functional Requirement Performance Requirement
	Ambiguous	Derive	M.1.6 - The IFS shall maintain ground clearance of X ft to allow maintenance.	Performance Requirement
11. For limit load cases, the IFS shall not permanently deform	—	—	S.1.2 - For limit load cases, the IFS shall not permanently deform.	Physical Requirement
	Ambiguous	Derive	M.1.7 - It shall be enunciated to the pilot when loads equal to or greater than the ultimate load are encountered.	Physical Requirement
12. For ultimate load cases, it shall be allowable that the IFS permanently deform	—	—	S.1.4.1 - For the ultimate load case, the IFS shall be allowed to permanently deform.	Physical Requirement
	Ambiguous	Derive	S.1.4.2 - Permanent deformation in the ultimate load case shall represent at most a hazardous failure.	Performance Requirement
13. The IFS shall be capable of performing its intended functions with one, latch, hinge, or bumper failed	Ambiguous	Add	W.1.3.1 - Redundant access support components shall contribute less than X to the GTOW.	Physical Requirement
			L.1.3 - The left and right IFS halves shall latch together at the X position.	Physical Requirement
	Derive		L.1.4 - Loading due to latch failure shall not exceed the ultimate load factor.	Physical Requirement
			L.1.5 - Each latching component shall be capable of failing and functioning independently.	Physical Requirement
			L.1.5.1 - Single latch failure shall not be catastrophic P.1.1 - The external aerodynamic surface drag count shall be less than or equal to X.	Performance Requirement Performance Requirement
14. The IFS shall have a mechanism or means of visible indication of a burst duct failure that is visible during a pilot walk-around	Ambiguous	Derive	M.1.8 - Burst duct failure shall be enunciated.	Functional Requirement
			M.1.8.1 - Burst duct failure shall be visibly obvious on pilot walkaround.	Physical Requirement
15. Seals shall be provided to the IFS design to ensure firewall integrity with adjacent structural components	Ambiguous	Add	W.1.5.1 - IFS seals shall contribute less than X to the GTOW.	Physical Requirement

		Derive	M.1.1.2 - The IFS shall use Inconel, which is considered fireproof, for all fireproof components and interfaces.	Design Constraint
	Compound	Split	I.1.1.3 - The IFS shall use seals to ensure firewall integrity with adjacent components. I.1.2 - The IFS interfaces shall prevent the spread of fire to other components.	Design Constraint Functional Requirement
	Ambiguous	Add	W.1.4 - The ventilation system shall contribute less than X to the GTOW.	Physical Requirement
16. The IFS shall have ventilation features as necessary to meet the ventilation requirements of X lb/s.	Compound	Split	S.1.3 - The IFS shall have a system for managing engine core ventilation. S.1.3.1 - The IFS ventilation system shall have a vent rate capacity of at least X.	Physical Requirement Performance Requirement
	Ambiguous	Derive	S.1.3.2 - The IFS design shall include fasteners to attach the ventilation system. S.1.3.3 - The ventilation system shall be capable of withstanding the limit load temperatures. P.1.3 - The ventilation system must support overall engine air flow rates of at least X lb/s. P.1.4 - The pressure loss through a vent shall not exceed ΔX_{psi} .	Physical Requirement Performance Requirement Performance Requirement Performance Requirement

Appendix C. Solution Domain Requirements Table

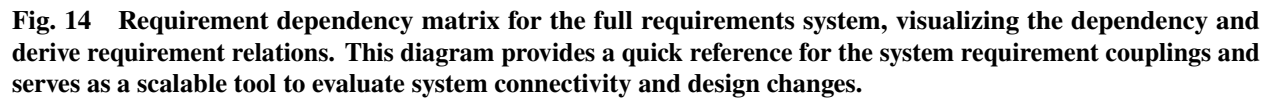
Table 6 The solution domain requirements table contains all 76 requirements from the eight design categories. This requirements space resulted from the normalization processes described in Table 2.

Item	Text
W.1	The IFS shall contribute less than X of the total GTOW.
W.1.1	The air washed surface shall contribute less than X to the GTOW.
W.1.2	The acoustic treatment shall contribute less than X to the GTOW.
W.1.3	The latching components and other access support structures shall contribute less than X to the GTOW.
W.1.3.1	Redundant access support components shall contribute less than X to the GTOW.
W.1.4	The ventilation system shall contribute less than X to the GTOW.
W.1.5	Fire control systems shall contribute less than X to the GTOW.
W.1.5.1	IFS seals shall contribute less than X to the GTOW.
W.1.5.2	Firewalls shall contribute less than X to the GTOW.
S.1	The IFS configuration and limit load capacity shall support both regular operation, and loads resulting from hazardous conditions.
S.1.1	The limit load factor shall be at least X.
S.1.2	For limit load cases, the IFS shall not permanently deform.
S.1.3	The IFS shall have a system for managing engine core ventilation.
S.1.3.1	The IFS ventilation system shall have a vent rate capacity of at least X.

S.1.3.2	The IFS design shall include fasteners to attach the ventilation system.
S.1.3.3	The ventilation system shall be capable of withstanding the limit load temperatures.
S.1.4	The IFS mounting shall maintain integrity under the ultimate load condition.
S.1.4.1	For the ultimate load case, the IFS shall be allowed to permanently deform.
S.1.4.2	Permanent deformation in the ultimate load case shall represent at most a hazardous failure.
S.1.5	Following engine failure, the IFS shall be able to support the windmilling loads for X.
S.1.6	The IFS shall maintain a positive margin under limit load conditions after exposure to fire for X.
S.1.7	The air washed surface shall conform to the inner bypass duct lines within X.
S.1.8	The air washed surface waviness shall be within X tolerance.
S.1.9	Each means of holding the IFS open shall be capable of supporting the access port independently.
S.1.9.1	Each means of holding the IFS open shall be capable of supporting at least X.
M.1	The IFS design shall support long-term durability and maintenance of the engine.
M.1.1	The IFS shall contain fire to the engine core environment.
M.1.1.1	Engine access points shall include a fire seal.
M.1.1.2	The IFS shall use Inconel, which is considered fireproof, for all fireproof components and interfaces.
M.1.1.3	The IFS shall be designed to survive fire for a minimum of 15 minutes.
M.1.2	The IFS design shall include mounting structures for the fire control systems.
M.1.3	The IFS shall have a means of accessing the engine for repair.
M.1.3.1	The IFS shall have a means of accessing the engine accessories for repair.
M.1.3.2	The IFS design shall include mounting structures for access ports and latches.
M.1.3.3	Blocker doors and drag links shall have features that enable removal for maintenance.
M.1.4	There shall be at least two means of holding the IFS open during maintenance.
M.1.5	The IFS shall include attachment points for general service equipment.
M.1.6	The IFS shall maintain ground clearance of X ft to allow maintenance.
M.1.7	It shall be enunciated to the pilot when loads equal to or greater than the ultimate load are encountered.
M.1.8	Burst duct failure shall be enunciated.
M.1.8.1	Burst duct failure shall be visibly obvious on pilot walkaround.
M.1.9	The inner IFS surface shall clear engine externals by at least X.
M.1.9.1	Based on stack-up and deflection analysis, the inner IFS structure clearance shall deviate within Y in.
M.1.9.2	The inner IFS surface and engine externals shall maintain clearance of at least +/-X tolerance at temperature.
I.1	IFS interfaces shall control engine flow and prevent the spread of hazards.
I.1.1	The IFS shall accommodate firewalls.
I.1.1.1	The IFS shall accommodate firewalls between the core and pylon.
I.1.1.2	The IFS shall accommodate firewalls between the core and the fan compartments.
I.1.1.3	The IFS shall use seals to ensure firewall integrity with adjacent components.
I.1.2	The IFS interfaces shall prevent the spread of fire to other components.
I.1.3	The aerodynamic interface between the fan and IFS shall be maintained within +/-X tolerance.
L.1	Latches shall provide access to the IFS.
L.1.1	Maintenance access shall be provided using latches.
L.1.2	The latching status shall be enunciated.
L.1.2.1	Latching status shall be within view during pilot walkaround.
L.1.2.2	All latch failures shall be enunciated on the ground.

L.1.3	The left and right IFS halves shall latch together at the X position.
L.1.4	Loading due to latch failure shall not exceed the ultimate load factor.
L.1.5	Each latching component shall be capable of failing and functioning independently.
L.1.5.1	Single latch failure shall not be catastrophic
L.1.6	Latches shall remain sealed under the limit load conditions for X min.
R.1	The IFS design shall have a clear path to certification.
R.1.1	The IFS design shall meet regulations for both traditional fuel and alternative energy systems.
R.1.2	The IFS shall satisfy all fire requirements as outlined by the FAA/EASA.
R.1.3	The IFS shall not depart the aircraft under any propulsion system failure scenario.
R.1.4	The IFS shall maintain airworthiness for at least X minutes in the ultimate load case.
N.1	The IFS shall at least meet regulatory requirements for noise reduction.
N.1.1	The air washed surface shall meet the minimum acoustic area requirements of X sq. in.
N.1.2	The acoustic treatment shall attenuate acoustic frequencies within a range of X-Y frequencies.
N.1.3	The acoustic treatment shall maintain at most X EPNdB during takeoff and landing events after an area loss of up to Y sq.in.
N.1.4	The acoustic treatment shall maintain at most X EPNdB during flight after an area loss of up to Y sq.in.
P.1	The IFS shall support optimal aircraft performance.
P.1.1	The external aerodynamic surface drag count shall be less than or equal to X.
P.1.2	Duct pressure loss shall not exceed ΔX psi.
P.1.3	The ventilation system must support overall engine air flow rates of at least X lb/s.
P.1.4	The pressure loss through a vent shall not exceed ΔX psi.

Downloaded by University of Michigan on October 22, 2026 | http://arc.aiaa.org | DOI: 10.2514/6.2026-4230



The below activity diagrams are representative of those developed in MagicDraw for this project. For additional model diagrams, please contact the authors.

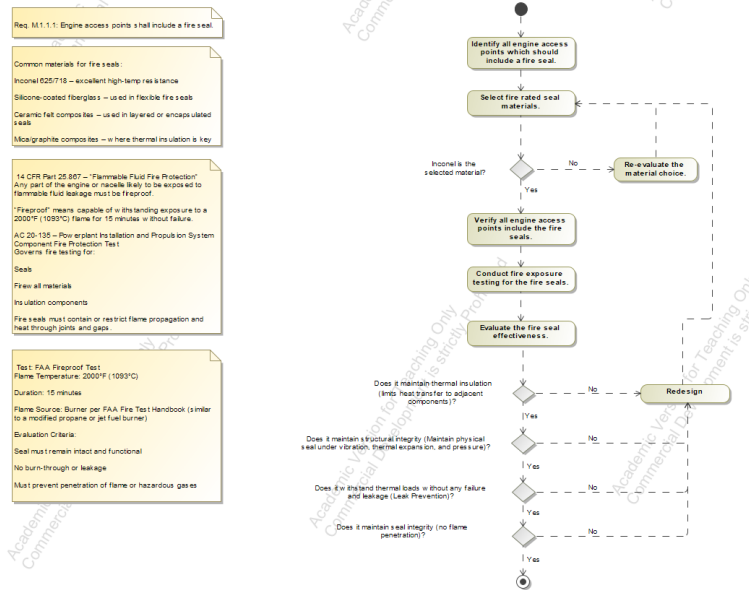


Fig. 15 Activity diagram for requirement *M.1.2.1*, a product requirement for the baseline IFS model.

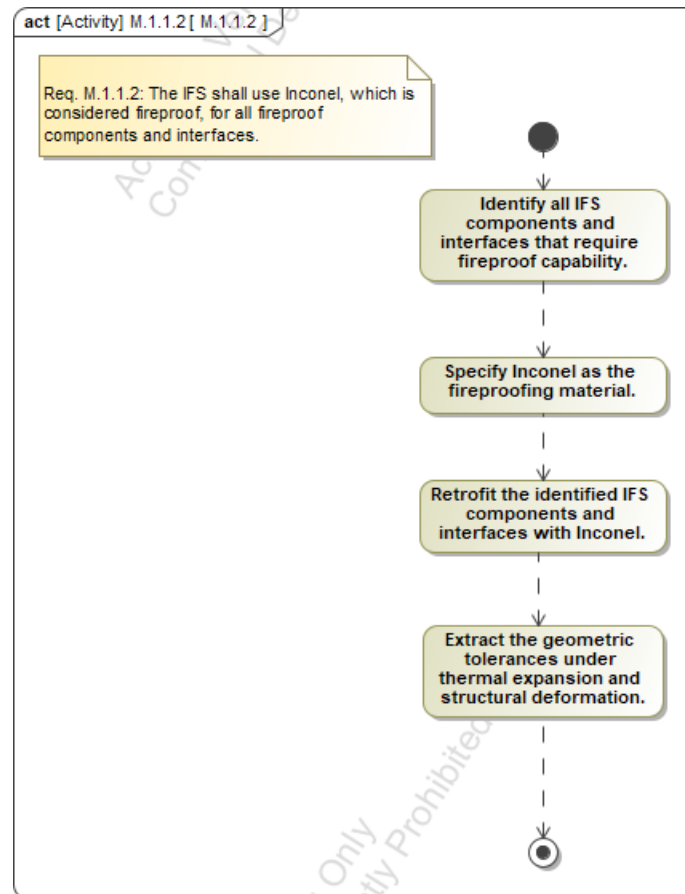


Fig. 16 Activity diagram for requirement *M.1.1.2*, a process requirement for the baseline IFS model.

Acknowledgments

The authors would like to thank Michael Harke, Jonathan Hartman, and Charlene Hu from Collins Aerospace, and Brian Wolhaupter from the University of Illinois Urbana-Champaign for their valuable support and technical feedback for the duration of this project.

This project received funding from Collins Aerospace under the award number AWD024885.

References

- [1] Saada, H., and Garcia Garriga, A., “A MBSE-based framework for aircraft systems design and evaluation,” *Aerospace European Conference 2023*, 2023, p. 298.
- [2] Saloranta, S., “From Documents to Models: Adopting MBSE Practices in Complex Embedded Systems and Leveraging Large Language Models,” Master’s thesis, University of Turku, 12 2024.
- [3] Oroz, J., Roohi, Z. A., Abelezele, S., Fronk, G., Fawares, R. A., Pinon-Fischer, O. J., Gharbi, A., Mavris, D. N., Petersen, M., Karl, A., Matlik, J. F., and Schwering, B., “Implementing the Digital Thread - A Proof-of-Concept,” *AIAA SCITECH 2023 Forum*, 2023. <https://doi.org/10.2514/6.2023-1405>, URL <https://arc.aiaa.org/doi/abs/10.2514/6.2023-1405>.
- [4] Seitz, A., Nickl, M., Stroh, A., and Vratny, P. C., “Conceptual study of a mechanically integrated parallel hybrid electric turbofan,” *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 232, No. 14, 2018, pp. 2688–2712. <https://doi.org/10.1177/0954410018790141>, URL <https://doi.org/10.1177/0954410018790141>.
- [5] Zhao, J., Zhang, X., Swaminathan, N., and Haran, K. S., “An Overview of High Specific Power Electrical Machines and Drives Technologies for Electrified Aircraft,” *2022 IEEE Energy Conversion Congress and Exposition (ECCE)*, 2022, pp. 1–8. <https://doi.org/10.1109/ECCE50734.2022.9948102>.
- [6] Jagtap, S. S., Strehlow, A., Reitz, M., Kestler, S., and Cinar, G., “Model-Based Systems Engineering Approach for a Systematic Design of Aircraft Engine Inlet,” *AIAA SCITECH 2025 Forum*, 2025, p. 1410.
- [7] Collins Aerospace, “Nacelle Systems,” , 2025. URL <https://www.collinsaerospace.com/what-we-do/industries/commercial-aviation/aerostructures/nacelle-systems>.
- [8] Aten, M., “Patent for thrust reverser with one or more buttressing corner cascade portions,” , 2015. URL <https://data.epo.org/publication-server/rest/v1.0/publication-dates/20151028/patents/EP2937549NWA1/document.html>.
- [9] Olson, T., Cobia, J., and Sommer, T., “Patent for thrust reverser inner fixed structure with corner fitting,” , 2014. URL <https://register.epo.org/application?number=EP14159512&tab=main>.
- [10] Cottrell, D. B., Mahmood Alwash, M. A., and Hennig, G. S., “Patent for integrated electric nacelle system and method,” , 2014. URL <https://patents.justia.com/patent/20220306305>.
- [11] Airbus, “Clean Aviation SWITCH Project to Advance Hybrid-Electric and Water Enhanced Turbofan Technologies,” , 2022. URL <https://www.airbus.com/en/newsroom/press-releases/2022-11-clean-aviation-switch-project-to-advance-hybrid-electric-and-water>.
- [12] Xie, Y., Savvarisal, A., Tsourdos, A., Zhang, D., and Gu, J., “Review of hybrid electric powered aircraft, its conceptual design and energy management methodologies,” *Chinese Journal of Aeronautics*, Vol. 34, No. 4, 2021, pp. 432–450. <https://doi.org/https://doi.org/10.1016/j.cja.2020.07.017>, URL <https://www.sciencedirect.com/science/article/pii/S1000936120303368>.
- [13] Yu, F., Chen, J., Gao, P., Kong, Y., Sun, X., Wang, J., and Chen, X., “A Review of Hybrid-Electric Propulsion in Aviation: Modeling Methods, Energy Management Strategies, and Future Prospects,” *Aerospace*, Vol. 12, No. 10, 2025. <https://doi.org/10.3390/aerospace12100895>, URL <https://www.mdpi.com/2226-4310/12/10/895>.
- [14] Duffy, K. P., and Jansen, R. H., “Turboelectric and Hybrid Electric Aircraft Drive Key Performance Parameters,” *2018 AIAA/IEEE Electric Aircraft Technologies Symposium (EATS)*, 2018, pp. 1–19.
- [15] Henderson, K., and Salado, A., “Value and benefits of model-based systems engineering (MBSE): Evidence from the literature,” *Systems Engineering*, Vol. 24, No. 1, 2021, pp. 51–66. <https://doi.org/10.1002/sys.21566>, URL <https://ideas.repec.org/a/wly/syseng/v24y2021i1p51-66.html>.

- [16] AIAA Digital Engineering Integration Committee, “Digital engineering workforce development: Challenges, best practices, and recommendations,” Tech. rep., American Institute of Aeronautics and Astronautics, 2025. URL https://aiaa.org/resources/digital-engineering-workforce-development-white-paper?_gl=1*cc4xq1*_gcl_au*Ntc2Mzk1MjQuMTc1NzYwNDM2OQ..*_ga*NjA2MjIyNTA3LjE3NDEzODA4NjM.*_ga_BFMKMMYM72*czE3NjI3OTgzMTIkbzUxJGcxJHQxNzYyNzk5OTcyJGo2MCRsMCRoMA.
- [17] American Institute of Aeronautics and Astronautics (AIAA), “How Digital Engineering Is Enabling Breakthrough Capabilities In Aerospace,” , 2023. URL <https://aviationweek.com/aerospace/how-digital-engineering-enabling-breakthrough-capabilities-aerospace>.
- [18] International Council on Systems Engineering (INCOS), “MBSE initiative,” , n.d. URL <https://www.incose.org/communities/working-groups-initiatives/mbse-initiative>.
- [19] Li, T., Lockett, H., and Lawson, C., “Using requirement-functional-logical-physical models to support early assembly process planning for complex aircraft systems integration,” *Journal of Manufacturing Systems*, Vol. 54, 2020, pp. 242–257. <https://doi.org/https://doi.org/10.1016/j.jmsy.2020.01.001>, URL <https://www.sciencedirect.com/science/article/pii/S0278612520300017>.
- [20] Friedenthal, S., Moore, A., and Steiner, R., *Practical Guide to SysML*, 1st ed., Morgan Kaufmann, Amsterdam; Boston, 2008. <https://doi.org/10.1016/B978-0-12-374379-4.X0001-X>, URL <https://linkinghub.elsevier.com/retrieve/pii/B9780123743794X0001X>.
- [21] “Systems Engineering Vision 2035,” , 2021. URL <https://sevisionweb.incose.org/>.
- [22] Miller, W., *The Future of Systems Engineering: Realizing the Systems Engineering Vision 2035*, IOS Press, 2022. <https://doi.org/10.3233/ATDE220707>.
- [23] Hutcheson, R., Mcadams, D., Stone, R., and Tumer, I., “Function-based systems engineering (FuSE),” *International Conference on Engineering Design 2007*, 2007.
- [24] No Magic Inc., Dassault Systemes, “MagicDraw Documentation: Relating Requirements,” , 2025. URL <https://docs.nomagic.com/spaces/CRMP190/pages/33194278/Relating+requirements>.
- [25] No Magic Inc., Dassault Systemes, “No Magic Documentation,” , 2024. URL <https://docs.nomagic.com/>.
- [26] Raymer, D., *Aircraft Design: A Conceptual Approach, Sixth Edition*, American Institute of Aeronautics and Astronautics (AIAA), 2018. <https://doi.org/10.2514/4.104909>.
- [27] Kratz, J. L., and Culley, D. E., “Enabling a Weight Efficient Power System for an Electrified Turbofan Through Gearbox Design and Control,” *AIAA SCITECH 2025 Forum*, 2024. <https://doi.org/10.2514/6.2025-1910>, URL <https://arc.aiaa.org/doi/abs/10.2514/6.2025-1910>.