

A Model-Based System Engineering Approach to Support System Architecting Activities in Early Aircraft Design

Nikta Tabesh

A Thesis

in

The Department of

Mechanical, Industrial and Aerospace Engineering

Presented in Partial Fulfillment of the Requirements for the Degree of

Master of Applied Science (Mechanical Engineering) at Concordia University

Montreal, Quebec, Canada

June 2023

© Nikta Tabesh, 2023

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Nikta Tabesh**

**Entitled: A Model-Based Systems Engineering Approach to Support System Architecting
Activities in Early Aircraft Design**

and submitted in partial fulfillment of the requirements for the degree of
Master of Applied Science (Mechanical Engineering)

complies with the regulations of this University and meets the accepted standards with respect to
originality and quality. Signed by the Final Examining Committee:

_____ *Dr. Onur Kuzgunkaya* Examiner & Chair

_____ *Dr. Catharine Marsden* External Examiner

_____ *Dr. Susan Liscouët-Hanke* Supervisor

Approved by _____

Dr. Martin D. Pugh, Chair

Department of Mechanical, Industrial and Aerospace Engineering

_____ 2023

_____ Dr. Mourad Debbabi, Dean

Gina Cody School of Engineering and Computer Science

Abstract

A Model-Based System Engineering Approach to Support System Architecting Activities in
Early Aircraft Design

Nikta Tabesh

The aviation industry aims to reduce its environmental footprint and meet ambitious environmental targets, prompting the exploration of novel aircraft concepts and systems, such as hybrid-electric or distributed propulsion. These emerging technologies introduce complexity to aircraft system architectures, requiring innovative approaches to design, optimization, and safety assessment, particularly for system architecting. Several aspects of system architecting specification and evaluation are typically performed separately, using different people and a mix of manual and model-based processes. Connecting these activities has the potential to make the design process more efficient and effective. This thesis explores how a Model-Based Systems Engineering (MBSE) specification environment can be structured and enriched to enable a better bridge to Multidisciplinary Design Analysis and Optimization (MDAO) and Model-Based Safety Assessment (MBSA) activities. The proposed MBSE approach focuses on enhancing system specifications, particularly for unconventional system architectures, which typically feature greater variability in early design stages. Using the ARCADIA/Capella MBSE environment, a multi-level approach is proposed to structure the system architecture specification. In addition, a catalogue of modeling artifacts is established to facilitate the development of various hybrid-electric system configurations. The MDAO link mechanism is demonstrated with an example from the collaborative AGILE4.0 project. Two test cases demonstrate the implementation of the approach: a hybrid-electric propulsion system and associated sub-systems for the overall approach and the landing gear braking system for the model-based Functional Hazard Analysis (FHA), as an example of an MBSA activity. Overall, this thesis helps improve the integration and collaboration between engineers working on MBSE, MDAO, and MBSA. This better integration will help to reduce the development time and risk. Therefore, the presented thesis contributes to a more efficient aircraft development process, enabling the industry to tackle the emerging needs of unconventional aircraft systems and their integration.

Acknowledgment

First and foremost, I praise and thank God for giving me the strength throughout all the challenging moments of completing this thesis.

I would like to express my deepest gratitude to my parents, Mina and Mehdi. Their unwavering support, love, and guidance have been my beacon throughout this journey. No words can express my love for my family, and this accomplishment would not have been possible without them.

My heartfelt thanks go to my husband, Saman, who has been my pillar of strength, my confidante, and my source of endless inspiration. His patience, encouragement, and unwavering faith in my capabilities have empowered me to overcome the challenges of this journey. I am eternally grateful for his love and support throughout this process.

I would like to express my sincere gratitude to Dr. Susan Liscouët-Hanke for welcoming me to her laboratory in 2020. Throughout my journey, her profound expertise in the world of Aircraft and her patient guidance has kept me on the right track, and I am immensely grateful for her continuous support and advice over the past three years. I am grateful for her understanding and cooperation during the challenges that I had as a new immigrant to Canada.

I thank all my fellow Air Systems Lab student colleagues Andrew Jeyaraj, Carlos Rodriguez and Vijesh Mohan for their contributions and support. I owe profound thanks to my friend Andrew, who supported me with enthusiasm. His intellectual contributions and substantial assistance have been invaluable in keeping me motivated and to the completion of this thesis.

I thank all who contributed and supported within the AGILE 4.0 MDAO-NextGen project at Bombardier. Special thanks go to Vincent Saluzzi and Alvaro Tamayo for their constructive advice and their expertise. Their practical insights and industry perspective have added immense value to this research, bridging the gap between academia and real-world application.

Completing this thesis has been a significant milestone for me. The support I have received from each of you has made it possible. I am truly privileged to have had you all by my side during this remarkable journey. Thank you all once again for your support, encouragement, and guidance.

To my beloved family.

Contents

List of Figures	viii
List of Tables	xi
Nomenclature	xii
1 Introduction	1
1.1 Background and Motivation.....	1
1.2 Aircraft Development Process and Need for New Methods and Tools	2
1.2.1 Systems Architecting Process	6
1.2.2 The need for system architecture consideration in conceptual MDAO and MBSA. 9	
1.3 Objectives and Scope of the Thesis.....	10
1.4 Organization of the Thesis	11
2 State of the Art	12
2.1 System architecting in early aircraft design	12
2.2 Model-Based System Engineering (MBSE)	15
2.2.1 Commonly used MBSE tools	17
2.2.2 ARCADIA/Capella.....	18
2.3 Systems Architecting Considerations in MDAO Workflow	26
2.4 Model-based safety assessment (MBSA).....	30
2.5 Summary and Gap Analysis.....	33
3 Methodology	35
3.1 Multi-level System Specification.....	36
3.1.1 Model Development.....	37
3.1.2 Levels of Detail in the Modeling Approach	40

3.1.3	Model Variants Adaptation & Management.....	44
3.1.4	Architectural Element Catalog & Reusability	46
3.1.5	Model Nomenclature Consistency	47
3.2	Connecting MBSE to MDAO	49
3.2.1	Model Data Extraction and Connection to AGILE 4.0 MDAO workflow.....	51
3.3	MBSE to MBSA Integration.....	56
3.3.1	FHA integration within the MBSE framework.....	57
3.3.2	FHA Principles & Application.....	58
3.4	Summary	63
4	Model Integration and Application.....	65
4.1	Test case 1: Multi-level system specification for the hybrid-electric propulsion system and associated subsystems	65
4.1.1	Hybrid-Electric Propulsion System	65
4.1.2	Primary Flight Control System – Yaw control	82
4.2	Test-Case 2: Model-based FHA for the Landing Gear Braking System.....	85
4.3	Summary	92
5	Conclusion and Future Work	93
5.1	Summary of Contributions and Discussion.....	93
5.2	Future Work	96
	List of Publications	98
	Bibliography	99
	Appendix.....	107

List of Figures

Figure 1- 1: Safety integration to the aircraft development process, adapted from [17] 5

Figure 1- 2: Stages in the system architecting process for design space exploration 7

Figure 2- 1: Viewpoints-based approach in ARCADIA, from [68] 20

Figure 2- 2: Architecture representations using the multi-level modeling approach, from [25] .. 22

Figure 2- 3: Engineering activities supported by Capella viewpoint extensions, from [69] 23

Figure 2- 4: Stages of the system architecting process for design space exploration 27

Figure 3- 1: Overview of the methodology in Capella and connections to other analysis environments 35

Figure 3- 2: Multi-level modeling approach and levels of detail within the MBSE framework .. 38

Figure 3- 3: L0 model aircraft-level model of hybrid-electric aircraft partial series/parallel configuration at the LA layer with detail level 0 42

Figure 3- 4: L1 model aircraft-level model of hybrid-electric aircraft partial series/parallel configuration at the LA layer with detail level 1 43

Figure 3- 5: L2 model aircraft-level model of hybrid-electric aircraft partial series/parallel configuration at the LA layer with detail level 2 for propulsion and electrical systems 43

Figure 3- 6: Horizontal adaptation for aircraft to system-level transition 45

Figure 3- 7: Proposed framework for integrating MBSE and MDAO for systems architecting, from [96] 50

Figure 3- 8: Tool selection for MBSE-MDAO integration framework, from [96] 52

Figure 3- 9: Algorithm of parameter input and extraction in Capella 53

Figure 3- 10: Integrating Capella with AGILE4.0 workflow, from [96] 55

Figure 3- 11: Overview of the formalized process of FHA integration and application within Capella 60

Figure 3- 12: Safety assessment property package defined in Viewpoint Editor by PVMT and applied to a system function at the SA layer in Capella	62
Figure 4- 1:Hybrid- electric propulsion architectures, extracted from [113].....	66
Figure 4- 2: Series hybrid powertrain, from [94].....	67
Figure 4- 3: Parallel hybrid powertrain, from [94]	69
Figure 4- 4: Series/Parallel hybrid powertrain with additional electric boost, from [94].....	70
Figure 4- 5: S0-Generic system architecture for hybrid-electric aircraft without interfaces systems at SA in Capella.....	71
Figure 4- 6: L0-Generic for Parallel HE configuration at LA in Capella.....	73
Figure 4- 7: L1 model for partial series/parallel HE configuration at LA in Capella.....	75
Figure 4- 8: Specification of systems illustrated in L2 and P2 models, adapted from [94], [117], [118].....	76
Figure 4- 9: L2 model for partial series/parallel HE configuration in Capella.....	78
Figure 4- 10: P2 model for partial series/parallel HE configuration in Capella	81
Figure 4- 11: Challenger 605 yaw control schematic, from [118].....	83
Figure 4- 12: Sys-L1 for FCS yaw control and interfaces to HE systems in Capella.....	84
Figure 4- 13: Integration of AFHA and SFHA scopes functional decomposition in [SFBD] and [LFBD] diagrams in Capella.....	85
Figure 4- 14: Allocation of aircraft-level functions to the system(s) in Capella at SA and LA modeling levels	86
Figure 4- 15: Allocation of system-level functions to the braking system with color-coding at [LAB] logical architecture diagram L2 model in Capella	87
Figure 4- 16: Functional failure effect relationships related to the brakes system functions using [SDFB] and [LFBD] Functional chain diagrams in Capella	88
Figure 4- 17: Safety properties allocation to the functions evident the failure relationship on ground	89

Figure 4- 18: Functional failure effect event and their relationships analysis potential using the functional scenarios [FS] diagram in Capella.....	90
Figure 4- 19: Flight phases and expected operating functions during a normal operation flight and rejected take-off in [MSM] diagram in Capella,	91
Figure A- 1: Sys-L1 for hydraulic system and interfaces to HE systems in Capella.....	107
Figure A- 2: Sys-L2 for FCS yaw control and interfaces to HE systems in Capella.....	108
Figure A- 3: Sys-P2 for FCS yaw control and interfaces to HE systems in Capella.....	109
Figure A- 4: Allocation of properties to battery physical component in Capella using the PVMT add-on	113
Figure A- 5: Condensed and expanded versions of the simplified Capella XML Schema	116
Figure A- 6: Capella XML data schema: the highlighted section shows where properties have been added manually.....	117
Figure A- 7: Activating PVMT add-on to a Capella project	118
Figure A- 8: Launching PVMT and specifying a viewpoint configuration by Viewpoint Definition Editor	119
Figure A- 9: Creating property domain in Viewpoint Editor	120
Figure A- 10: Defining enumeration literal as parameters of properties	121
Figure A- 11: Defining the scope of the property domain and adding more parameters as properties.....	122
Figure A- 12: defining different property types for the extensions	123
Figure A- 13: Safety viewpoint created using PVMT plugin in Capella to support model-based FHA.....	124

List of Tables

Table 3- 1: Levels of detail specification in the Capella modeling methodology	40
Table 3- 2: Capella nomenclature of elements in different modeling layers for the propulsion system	48
Table 3- 3: Overview of selected Capella diagram in support of implementing the FHA process	61
Table A- 1: Propulsion and electrical systems nomenclature of model components in Capella	110
Table A- 2: Hydraulic and fuel system nomenclature of models components in Capella.....	111
Table A- 3: Primary flight control system nomenclature of model components in Capella	112
Table A- 4: Mapping between Capella schema and simplified schema tag names	114

Nomenclature

AFHA	Aircraft Functional Hazard Analysis
APU	Auxiliary Power Unit
ARCADIA	Architecture Analysis and Design Integrated Approach
ARP	Aerospace Recommended Practice
ASSET	Aircraft System Sizing Estimation Tool
ASTRID	Aircraft On-board Systems Sizing and Trade-off Analysis in Initial Design
ATA	Air Transport Association
CAD	Computer-Aided Design
CMDOWS	Common MDAO Workflow Schema
CPACS	Common Parametric Aircraft Configuration Schema
CSV	Comma-Separated Values
EDP	Engine-Driven Pump
EMF	Eclipse Modeling Framework
EMP	Electric Motor Pump
FCS	Flight Control System
FHA	Functional Hazard Analysis
FMEA	Failure Mode and Effects Analysis
FS	Functional Scenario
HE	Hybrid-Electric
HTML	Hypertext Markup Language
JPL	Jet Propulsion Laboratory
LAB	Logical Architecture Breakdown
LA	Logical Architecture

LDFB	Logical Data Flow Blank
LFBD	Logical Functional Breakdown Diagram
L0	Aircraft-level Logical Architecture at Level 0
L1	Aircraft-level Logical Architecture at Level 1
L2	Aircraft-level Logical Architecture at Level 2
MBSA	Model-Based Safety Assessment
MBSE	Model-Based Systems Engineering
MDAO	Multidisciplinary Design Analysis and Optimization
MDO	Multidisciplinary Design Optimization
MSM	Modes and State Diagram
OA	Operational Analysis
OEM	Original Equipment Manufacturer
P2	Aircraft-level Physical Architecture at Level 2
PA	Physical Architecture
PAB	Physical Architecture Breakdown
PASA	Preliminary Aircraft Safety Assessment
PIDO	Process Integration and Design Optimization
PDU	Power Distribution Unit
PSSA	Preliminary System Safety Assessment
PVMT	Property Value Management Tool
RPL	Replicas
REC	Replicable Elements Collection
RTO	Rejected Takeoff
SA	System Analysis

SDFB	System Data Flow Blank
S0	Aircraft-level System Architecture at Level 0
SFBD	System Functional Breakdown Diagram
SFHA	System Functional Hazard Analysis
SoS	System of Systems
Sys-L1	System-level Logical Architecture at Level 1
Sys-L2	System-level Logical Architecture at Level 2
Sys-P2	System-level Physical Architecture at Level 2
SysML	Systems Modeling Language
TTR	Through the Road
UML	Unified Modeling Language

1 Introduction

The aviation industry has seen an increase in passenger numbers and aircraft operations frequency, raising many environmental concerns in the last decade [1]. The resurgence of passengers after the slowdown from the COVID-19 pandemic is expected to reach pre-pandemic levels in due course [2], [3]. Aviation still accounts for 2.8% of CO₂ emissions [4], and the industry aims to reduce its environmental impact by committing to reducing net global aviation carbon emissions by 50% by 2050 relative to the carbon emissions produced in 2005 [5], [6]. Reducing carbon emissions is a significant step towards achieving sustainability in the aviation sector. This has led to the exploration of novel aircraft concepts and systems, such as hybrid-electric and distributed propulsion, which promise to deliver a new generation of efficient and environmentally friendly aircraft. These emerging technologies introduce complexity to aircraft system architectures, requiring innovative approaches to design, optimization, and safety assessment [7], [8]. This research explores (1) a Mode-Based System Engineering (MBSE) approach for novel aircraft system architecture design, (2) a bridge to Multidisciplinary Design and Optimization (MDO) environment for aircraft system sizing, and (3) an integration to Model-Based Safety Assessment activities for system evaluation at the conceptual design stage. This thesis is conducted in collaboration with the Bombardier Inc., Canadian industrial partner in the European Union-funded project “AGILE4.0: Towards cyber-physical collaborative aircraft development” (2019-2023) [9]. This chapter presents the background and motivation for the conducted research, followed by the objectives and the scope of this thesis.

1.1 Background and Motivation

Aircraft manufacturers are focused on exploring new technologies to help meet environmental, emissions, and sustainability targets. Furthermore, adopting new technologies also gives manufacturers an advantage in a competitive market. These new technologies focus on electrifying aircraft systems and propulsion system architectures. One of the potential solutions is to use electricity to power aircraft for thrust production, system loads, or a combination of both. While fully electric aircraft is a promising technology, it requires further development of battery

technologies, and the energy density of electric power sources is lower than fuel, making it difficult to achieve the same endurance. Hence, Hybrid-electric (HE) aircraft combines the advantages of conventional and all-electric propulsion systems while maintaining high-performance capabilities and is seen as an intermediate step toward fully electric aircraft [10], [11]. HE propulsion and distributed electric propulsion system architectures, X-57 NASA [12], are being explored in the industry as Pratt and Whitney project 804 [13]. Hydrogen electric aircraft have been studied in academia, and recent test flights of hydrogen fuel cell-powered aircraft have shown promising results [14]. Therefore, the trend towards developing hybrid-electric aircraft is expected to continue to grow as technology advances and becomes more efficient, cost-effective, and widely adopted.

Studies have also been conducted on larger aircraft models that incorporate hybrid-electric propulsion systems, such as the Boeing SUGAR Volt [15], NASA STARC-ABL, and ES Aero ECO-150 [16]. In addition, smaller aircraft, such as regional and commuter aircraft that carry 19-30 passengers on routes up to 500, are considered to be suitable for hybridization [17]. The focus of recent studies has been to assess the effects of hybridization on these aircraft and to determine the advantages of different types of hybrid electric propulsion system architectures. The DO-228 has been used in conceptual studies and as a technology testbed for hybridization in this aircraft category [18].

However, these new technologies feature complex interactions, especially in the case of electrified propulsion architectures. Ultimately, aircraft manufacturers need to ensure that they can reduce their time to market for new aircraft to remain competitive. Thus, the aircraft development process needs to be adapted to deal with the additional complexity inherent in new technologies. Therefore, new methods and tools are required to help design and develop aircraft featuring these technologies.

1.2 Aircraft Development Process and Need for New Methods and Tools

The aircraft development process typically consists of three stages: conceptual, preliminary, and detailed design [19]. In conceptual design, the top-level aircraft requirements derived from market and stakeholder studies are used to obtain the aircraft's initial weights and performance metrics.

At this stage, decisions about the initial aircraft concept are made, such as the selection of the aircraft configuration and technological decisions. The conceptual design phase of aircraft design is important as it allows for exploring various design options and evaluating practical considerations regarding aircraft integration and manufacture. Investing more effort into this phase can help reduce rework and costs in later design stages. Optimal design choices must be made in this phase, and tools for early evaluation of aircraft concepts should be adopted.

Multidisciplinary Design Analysis and Optimization (MDAO) is becoming more prevalent in the industry for both exploring innovative aircraft designs and improving conventional designs. MDAO is now a widely adopted tool during the conceptual design stage of aircraft development [20]–[22]. This thesis uses an MDAO framework to analyze the aircraft to determine its key performance parameters. Aircraft configurations are evaluated based on performance and the ability to meet stakeholder requirements.

A step further in the design process is the preliminary design, where the aircraft design is expanded, and the system architectures such as the flight control system, landing gear system, environmental control system, and other systems are developed further. During this phase, the specification is refined to include more detailed requirements and characteristics, such as the aircraft's size, weight, range, speed, and payload capacity. The preliminary design phase also includes the selection of major systems and components, such as the engine, avionics, and landing gear. Finally, in detailed design, component-level design takes place. The specification is further refined and expanded to include detailed design drawings, schematics, and engineering analyses. The detailed design phase involves the development of detailed engineering plans for the aircraft, including the design of individual components and subsystems. This is followed by system integration, testing, and manufacturing from the end of detail design.

The overall aircraft development process follows a rigorous systems engineering approach that is prescribed by the SAE Aerospace Recommended Practice ARP4754 [23] guideline “Development of Civil Aircraft and Systems.” Safety is held as paramount in the development process, and the ARP4761 [24] standard prescribes a detailed safety assessment process that happens in parallel with the development of the aircraft and its systems. Therefore, the aircraft development process will expect detailed consideration for safety and systematic design to integrate new technologies according to the ARP4754 [23]. Identifying safety concerns through the formal safety assessment

process necessitates modifications to the architecture or configuration of the aircraft. This can result in significant costs and penalties associated with development time. Thus, with the added complexity of hybrid and distributed electric propulsion architectures, it is important to capture aspects of safety and systems architecting in early design stages, such as conceptual design. By addressing safety concerns at each level of the design process, the safety features are incorporated throughout the entire design of the aircraft and its systems.

The V-model in Figure 1-1 shows the integration between the safety and aircraft development process in a simplified manner, adapted from [23]. Requirements are organized into three levels: aircraft level, system level, and component level. The aircraft level requirements pertain to the overall performance and characteristics of the aircraft, such as payload, range, speed, and weight. At this level, safety requirements are generated from the aircraft Functional Hazard Assessment (FHA) based on the functions of the aircraft. The FHA is a systematic process that identifies potential hazards associated with the aircraft's functions and assesses the risk associated with each hazard. Safety requirements are also generated from other safety analyses listed in Figure 1-1.

System-level requirements are derived from aircraft-level requirements and are specific to the design of individual systems. System-level safety requirements are generated from the system FHA that are decompositions of the aircraft-level safety requirements. Component-level requirements are further derived from system-level requirements and influence the design or selection of individual components that make up the system. Safety requirements at this level are all aspects of the individual components that contribute to meeting the safety objectives associated with the system FHA classifications.

Validation is performed between each level of requirements to ensure that the correct system and specifications are being developed. Once the system is integrated, its design is verified against the requirements at each level to ensure that it has been built correctly and meets the specifications. As the design process progresses, more detail is added to the aircraft specification at each stage. This thesis focuses on aircraft development and safety integration activities during the conceptual design stage.

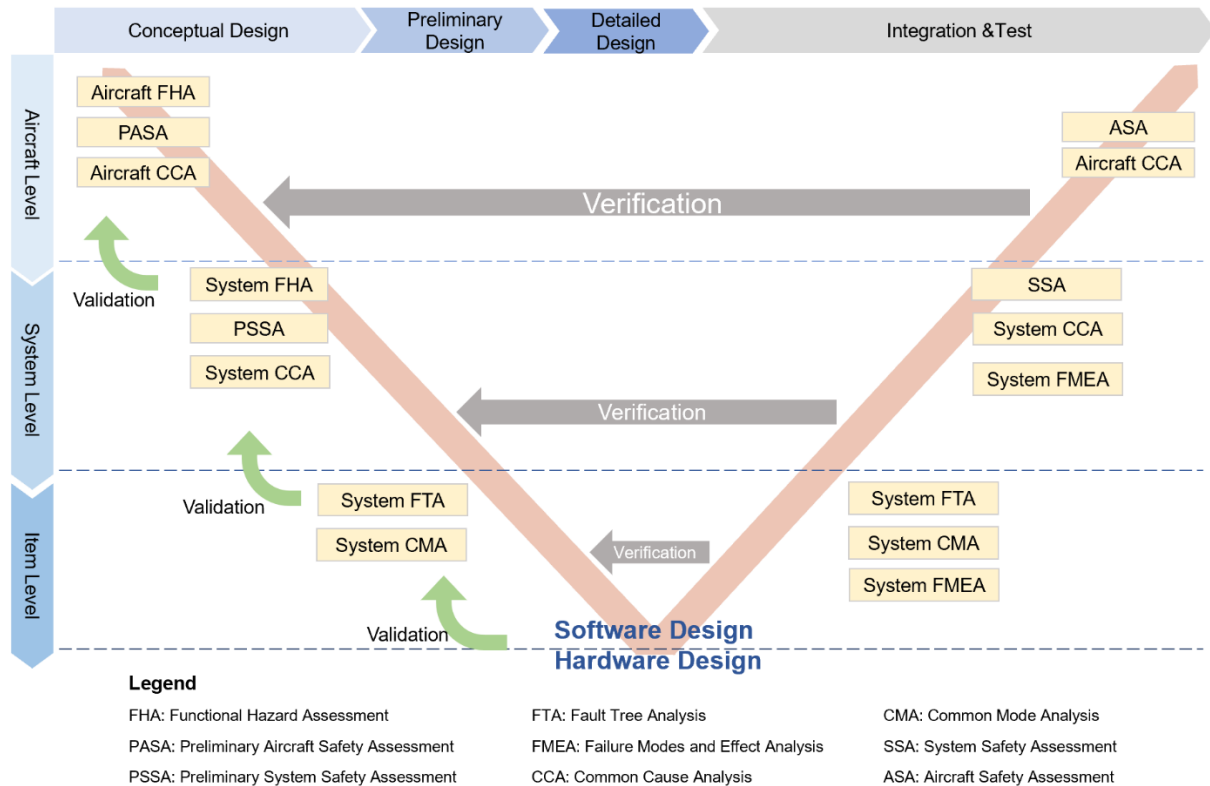


Figure 1- 1: Safety integration to the aircraft development process, adapted from [17]

In classic conceptual design methods, the system architecture and its impact on the overall weight and performance of the aircraft are not directly considered. Typically, a system architecture is fixed based on experience from past aircraft programs or supplier data, and the impact of the architecture is evaluated using semi-empirical approaches [25]. Recent developments have resulted in more advanced systems architecting techniques that allow the architecture to be used as input to the conceptual level MDAO processes. Some of these methods have been developed as part of the AGILE4.0 project, where aspects of system architecture, certification regulation, and safety have been considered in a conceptual MDAO framework [26]. The objective of the AGILE4.0 project was to develop the next generation of Multidisciplinary Design and Optimization (MDO) processes for aircraft, with the aim to significantly decrease aircraft development costs and time-to-market, yielding greener, more affordable aircraft solutions. The AGILE Paradigm has been established as a blueprint for MDO, promoting collaboration among design teams with diverse expertise [22]. The general idea behind the AGILE MDAO workflow is to allow people from different organizations and specializations to link their tools to MDAO.

The results of these analyses could result in changes being made to the system's architecture to meet the aircraft's high-level requirements.

1.2.1 Systems Architecting Process

The system architecting is the process of characterizing a system by analyzing its components and interactions to address a particular requirement, as outlined by the system specifications. The output is a conceptual model of the system, which details its components, interactions, and ability to accomplish a series of functions. According to [27], system architecting is defined as “activities of defining, documenting, maintaining, improving, and certifying proper implementation of an architecture.” In essence, the system architecting process involves making decisions that define a system architecture and its subcomponents. These decisions typically rely on a combination of quantitative and qualitative criteria to assess the feasibility, readiness, and capability of the resulting system architecture to satisfy requirements [28]. The system architecture is crucial for designing and evaluating novel aircraft concepts, especially those that involve hybrid-electric, distributed propulsion, and advanced or more electrical subsystem architectures. Here, the system architecting process is subdivided into three sub-activities, according to [29], as shown in Figure 1-2:

1. System architecture definition
2. System architecture representation
3. System architecture evaluation

System architecture definition refers to the process of designing and specifying the structure and behavior of a system. It involves identifying and defining a system's various components and subsystems, their interconnections and interfaces, and the functions they perform. System architecture definition also includes defining the overall system requirements, constraints, and performance objectives.

System architecture representation is how the design of system architecture is documented, communicated, and visualized. It involves creating a graphical or textual model that captures the key elements and relationships of the system architecture, including its components, functions, interfaces, and interactions. Architecture representation carries information about different aspects

of the design process, such as sizing, technological choices, and system specifications. An illustration that focuses on decomposing a system into its constituent parts enables the assessment of safety and redundancy factors. In brief, architectural representation eliminates vagueness in the definition of system architecture, fosters a common comprehension of the architecture, and creates valuable data about the system for employment in the design process.

System architecture evaluation can be defined as the process of assessing and analyzing a system architecture design to determine whether it meets the system requirements and objectives. It involves examining the architecture from various perspectives, such as functionality, performance, reliability, safety, maintainability, and cost. System architecture evaluation aims to identify any weaknesses, risks, or opportunities for improvement in the design and ensure that the architecture is feasible, efficient, and effective in meeting the system requirements.

The system architecture process serves as a blueprint for the development of the system, providing a framework for the design, implementation, testing, and maintenance of the system throughout its lifecycle. Usually, system architectures are filtered, created, and examined using different forms of representation such as model-based system engineering artifacts, 2D drawings and schematics, and 3D models that show the internal arrangement of components in the system architecture.

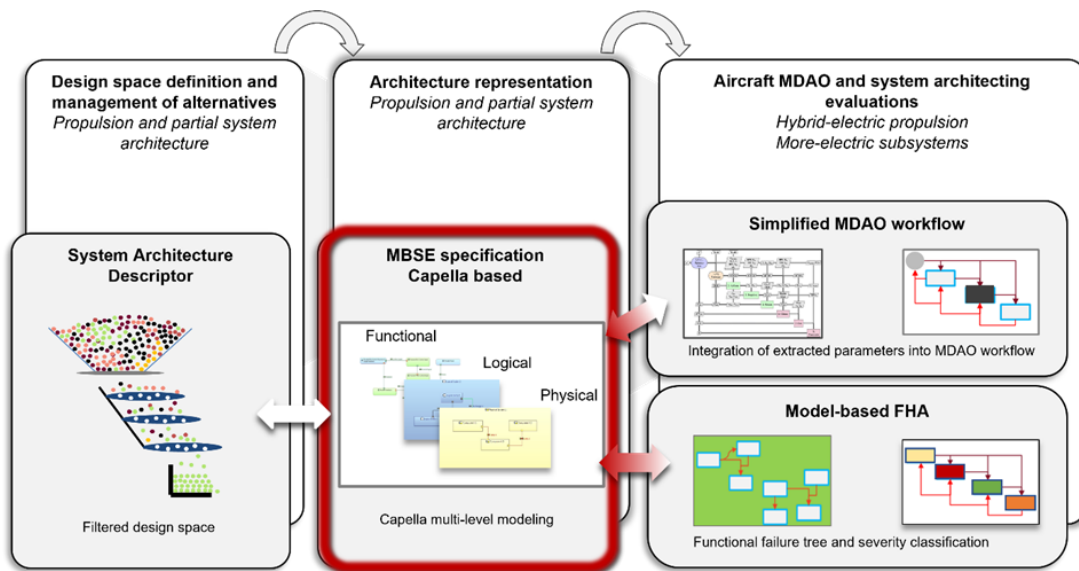


Figure 1- 2: Stages in the system architecting process for design space exploration

The current system architecting process is not based on a unified source of information; it uses a mix of manual and model-based approaches. Therefore, the traceability when dealing with system

design changes is limited, which can then impact the overall time and cost-effectiveness of system architecture. To advance the efficiency and effectiveness of system architecture, a central model-based specification approach is seen as promising to drive system development. This central model-based specification should capture all aspects of aircraft system architecture and support different viewpoints and analyses throughout the development process. There is a trend to improve the current system architecting process. As part of the AGIE 4.0 project, Boggero et al. [30] try to address the shortcoming of the traditional system architecting method by presenting a model-based approach.

A model-based approach provides characteristics and capabilities to ensure the system's development remains traceable, and it facilitates the generation of various system perspectives, comprehensive reports, and assorted documentation. Most importantly, this methodology enables tracking and accurate reflection of modifications in the system design within the system model. Emphasizing the representation of an aircraft system within its conceptual design, the architectural representation requires particular characteristics to effectively handle its inherent complexity. These imperative attributes include clarity for comprehensible understanding, traceability for accountable changes, systematic organization for logical structure, extensibility for future scalability, assessment capabilities for effective evaluation, encapsulation for maintaining the integrity of system components, and visualization for a clear, graphical representation [31].

Model-Based System Engineering (MBSE) is an advanced development methodology becoming more prevalent in the aeronautical industry [32]. It is seen as a solution to the complex system architecting process by allowing for more efficient capture of the technical design process. Using a common reference system model, MBSE can drive the system architecture, increase process coherence, and improve traceability throughout the system development. Mainly, MBSE applications in system architecting are in preliminary design [33], [34].

However, MBSE should be ideally used during the conceptual design phase, when the aircraft's requirements, system architecture, and feasibility are being conceptualized [29], [35]. During this phase, the overall vision and concept of the aircraft are established as limited information about subsystems is available, allowing greater design flexibility. By introducing MBSE during the conceptual design phase, the development team can create a system model that captures the system's requirements, functional behavior, and architecture. This model serves as a central

knowledge repository and provides a holistic view of the system, enabling better communication and collaboration among the stakeholders. Many of the most important design decisions are made during conceptual design, making it an ideal time for system architects to develop modeling artifacts to capture system requirements in the form of an architecture that can be updated as the design progresses through subsequent stages.

1.2.2 The need for system architecture consideration in conceptual MDAO and MBSA

One of the main difficulties in the current aircraft development processes is that MDAO and MBSE are carried out as separate activities that use distinct tools and require specific knowledge. Despite this, both processes aim to translate customer requirements into viable solutions. If a more comprehensive assessment of aircraft systems architecture is integrated into future MDAO frameworks, the specification of these architectures should originate from a systematic MBSE process. Additionally, it is preferable that the architectures that are examined and assessed during the conceptual design phase are smoothly transitioned into later design stages to minimize expenses and avoid unnecessary modifications.

Regarding safety assessment, the current approach still heavily depends on elaborate safety assessment procedures and does not generate a system architecture specification model that incorporates the findings of previous safety analyses. It is necessary to conduct more of the safety processes recommended by ARP 4761 during conceptual design and combine the results of these analyses in a system architecture model. This model can then be used during preliminary design, where the most recent advances in Model-Based Safety Assessment (MBSA) can be utilized to carry out detailed safety assessment activities as the architecture specification evolves.

In summary, the aircraft development process can be improved by exploring aircraft systems architectures early on. However, a key gap in this process is the lack of an efficient means for architecture representation and visualization. A formalized representation of systems architectures is necessary throughout development to improve efficiency. The development of an architectural representation framework for the conceptual design process can enable the efficient exploration of systems architectures. Formal architecture modeling enables important analysis, such as safety assessment, FHA, and MDAO activities, early in the design process. This investigation of a broad

range of system architectures leads to the development of more efficient aircraft. Clear architecture representation also identifies integration issues early, preventing costly program delays and making the product more competitive.

1.3 Objectives and Scope of the Thesis

This thesis focuses on the system architecture representation within the MBSE environment and integration to the evaluation phases, i.e., a simplified MDAO workflow and FHA early in conceptual design, as highlighted in Figure 1-2. This scope addresses system architecture representation for hybrid-electric propulsion, secondary power generation, and distribution specifications by a generic template model adaptable for various aircraft configurations. As a higher level of integration is required between the propulsion and non-propulsive power generation for various aircraft systems, including primary flight control system, and hydraulic and fuel systems, hybrid-electric is an example of sufficient complexity to illustrate the methodology.

The architectures of different hybrid-electric propulsion system configurations are created based on predefined requirements and aircraft configurations. The objectives of this research are as follows:

- Develop a generic set of architecture representations and modeling elements within the open-source Architecture Analysis & Design Integrated Approach (ARCADIA)/Capella MBSE framework to support the specification and analysis of aircraft with unconventional propulsion systems to manage architecture complexity and variability. This thesis focuses on hybrid-electric propulsion systems and associated aircraft subsystems.
- Explore how to link an MBSE specification model and an associated MDAO workflow to enable data consistency between the two.
- Explore how the MBSE specification model can be used to perform safety assessment (following the SAE ARP4761). This thesis focuses on the development of a methodology in Capella to perform a model-based FHA.

1.4 Organization of the Thesis

The organization of this thesis is as follows: Chapter 2 presents an overview of the current state of MBSE tools and practices in aircraft system architecture representation and outlines the need to link MBSE and MDAO and the importance of safety considerations to enhance the system architecting during the conceptual design phase. Chapter 3 introduces a multi-level modeling methodology using ARCADIA/Capella to represent system architectures. It is followed by the MBSE-MDAO link to enhance the system architecting process and the integration of FHA within the MBSE environment. Chapter 4 provides examples of how this methodology is applied to hybrid electric system architectures and yaw control in primary flight control systems and the application of model-based FHA to the landing gear braking system. Finally, Chapter 5 concludes with a summary of the main points and proposes areas for future research.

2 State of the Art

This chapter presents the state of the art in aircraft systems architecting. It introduces Model-Based Systems Engineering (MBSE) applications and discusses prior work on bridging MBSE to the other system evaluation processes, including MDAO workflow and MBSA activities. Finally, it closes with a gap analysis.

2.1 System architecting in early aircraft design

System architecture development involves creating various configurations for a particular aircraft system. This is achieved through a process including listing, presenting, and evaluating all possible architectural design options. During this process, all combinations of system components are identified to generate a list of potential system architectures. Then, feasible system architectures are evaluated, and the most suitable solution architecture that meets the requirements is selected. As briefly discussed in Chapter 1, system architecting is divided into three separate phases, depicted in Figure 1-2. The main focus of this research and the related literature is on the design space representation and potential links to the evaluation process. Each phase is described as follows:

1. Design Space Definition

During this stage, the boundaries of the design space are established, including factors such as weight, performance metrics, high-level aircraft requirements, and technology selections. In the case of hybrid-electric aircraft, the design space is defined by power distribution, power transmission, and the propulsion system's efficiency [36]. These combinations may be technologically unfeasible, inconsistent, difficult to integrate, or have poor performance. Thus, part of the design space definition involves eliminating infeasible configurations before proceeding to the architecture evaluation phase.

As the MBSE framework developed in this thesis should be able to easily connect with architecture descriptions and representations, which are outputs of the design space definition phase referring to Figure 1-2, a brief review of literature about design space definition is presented here.

Basically, various techniques and approaches are used to define an architecture design space. A design space is the set of all possible configurations of a system architecture that can be created by combining different variations of components and interconnections. The function-based approach is used by Liscouët-Hanke [37] and Lammering [38] to define architecture within a system sizing and performance estimation framework. Bussemaker et al. in [39] offer a new method for modeling the system architecture design space in a way that yields a semantic representation using architecture design space graphs. These graphs provide a semantic representation of the architecture design space, including function-component mapping, component characterization, and component connection.

As the design space grows, it becomes important to filter feasible architecture configurations. Jeyaraj et al. [40] present a rule-based safety filtering approach for large design spaces in their safety-focused systems architecting framework. The safety-based approach introduced by Jeyaraj in [28] enhances the system architecture definition phase by introducing a rule-based safety filtering method for conventional and novel system architectures (i.e., for more electric, hybrid-electric, and distributed electric aircraft). This method allows the extraction of feasible architectures from a large design space automatically. This literature was conducted concurrently with the development of this thesis.

2. Design Space Representation (Architecture Representation)

Aircraft systems are complex, with many subsystems that interact with one another. Architecture representation is necessary to present the system interfaces and exchanges between system components and other system elements in an unambiguous way. Creating an architectural representation of the aircraft system enables a shared understanding of the system between all parties involved in the design process [31].

An initial set of users' needs is translated into textual requirements. These requirements derive a functional architecture specification for the system of interest. From this point on, two approaches can be taken. First, functional architecture can constrain the design space and generate many architectural options, as demonstrated in [41]. These architectures are then represented in a graph-based environment where safety rules are applied, and feasibility checks are conducted to remove

incompatible options. The second approach would be directly generating the list of candidate system architectures in a graph-based representation environment. The subsequent application of rules and filtering of architectures is the same as previously described. The remaining few architectures are then developed into a model-based specification using any of the numerous MBSE workbenches that are currently available [42]. The recent work by Boggero et al. [30], within the AGILE 4.0 project, explores a tool-independent approach and introduces a model-based architectural framework for complex system representation. This MBSE framework focuses on the system architecting activities of a systems engineering product development process, which includes functional, logical, and physical system architecting. Other MBSE approaches literature are explored in further detail in Section 2.2.

3. Architecture Evaluation

In early works on system architecture evaluation [37], [43] and later [37], [44], the overall system architecture contribution to aircraft level performance is evaluated (calculating contributions to weight, drag, and secondary power off-takes). This evaluation workflow can be generic and repeatable with minor modifications for conventional and even more electric system architectures [44]–[46]. However, reconfiguring the workflow may be time-consuming for unconventional and novel architectures, as models need to be identified, sizing and performance routines synthesized, and a link to aircraft-level metrics needs to be made. At the same time, aspects of tool fidelity must also be considered. At the conceptual design phase, obtaining overall results using a workflow consisting of low-fidelity tools may prove feasible. However, the effect of infusing some high-fidelity analysis as part of the workflow on aircraft-level metrics would still need to be quantified in terms of uncertainty. Although several approaches to formalizing the evaluation phase of system architecting have been proposed in the literature [38], [47], [48], these methods are based on an overall architecture description. They are sometimes very abstract, and the step of architecture representation is skipped, with the notable exception of [49], where system safety and performance analysis has been shown in the context of system architecting.

A more granular system architecture description and representation are required to perform more detailed system architecture evaluations. This can be achieved by using a so-called architecture descriptor. Typical descriptors may be textual, as in [44], pictorial, as in [46], and object-oriented, as in [41]. An architecture descriptor carries information about the system configuration that is

then used to activate specific modules and interfaces in a subsequent evaluation workflow. It is possible to evaluate many types of architectures based on structuring the architecture descriptor, but this ultimately limits the scope for applying the architect's knowledge base. This means that a system architect may not be able to easily add redundancies, change power types and incorporate new technologies without having to modify a baseline workflow each time. The graph-based system architecture descriptor proposed by Jeyaraj [28] is designed to support the integration of the safety-focused systems architecting framework within an MDAO environment and was developed in parallel to this thesis. This descriptor links architecture definition and rule-based safety assessment with architecture evaluation and formal architecture specification in an MBSE environment.

2.2 Model-Based System Engineering (MBSE)

MBSE is applied by systematic use of models that describe the behavior, functionality, requirements, activities, mission capability, structure, and interconnectivity between elements in the System of Systems (SoS) [50]. It enables the creation, organization, and distribution of data that is relevant to the developmental phase of a system engineering approach [51]. Given the complexity of system development and the need for extensive communication between diverse teams that are often in different geographical locations, MBSE has become essential for managing the large amounts of information generated during the process.

MBSE, as explained in [52] and [53], brings several benefits, such as improved communication, decreased risk of development, enhanced quality and safety, and elevated efficiency. Through a model-based process, the system model is interpreted structurally by all involved parties, leading to more effective communication and eliminating ambiguity in the design process. This is possible because the system model is communicated using a systematic modeling methodology, facilitating communication between design teams[53]. These models support visualization, validation, and verification during the lifecycle of a project. [50]. However, MBSE is not limited to a specific process and ensures that all the information related to the system design is stored in a model repository, resulting in a more uniform and standardized development process.

In [54] and [55], an overview of current MBSE methodologies and their application in different industries can be found. There is a growing interest in using MBSE for complex systems engineering, such as space systems architecting, due to its ability to provide a systematic modeling approach. In [56], MBSE is adopted for the preliminary design stage for satellite communication system architecting. Practicing individuals who use MBSE have said that it provides them with a clear understanding of design problems, helps them develop requirements, and speeds up the concept design process [35]. The Jet Propulsion Laboratory (JPL) has implemented MBSE with the aim of enhancing the quality of its products and missions, as well as reducing costs [57]. MBSE has been used to explore the trade space for fractioned satellite architectures at JPL [57] and [56]. Thus, MBSE is extensively used for developing highly integrated systems in various domains. The next section will explain how MBSE is implemented in aircraft system architecture representation.

MBSE has been increasingly adopted in the industry by Original Equipment Manufacturers (OEMs) such as Rolls Royce and Boeing, who have reported the benefits of using this approach for designing products [55]. In fact, Boeing has implemented MBSE in developing digital aircraft networks, resulting in reduced development time and the ability to identify design errors at an early stage [58]. Additionally, MBSE has been acknowledged as a methodology for handling the growing sophistication of automotive technology and design [59].

MBSE has been applied in various aircraft system development projects. Mathew et al. [60] used MBSE tools to develop a system architecture specification for integrated modular avionics, while Fisher et al. [61], Becker, and Giese [62] applied MBSE in aircraft system architecture specifications for small unmanned air vehicles and aircraft environmental control systems, respectively. Liscouet-Hanke et al. used MBSE to develop aircraft flight control systems [63], and Malone applied MBSE in the development of digital aircraft networks [58].

MBSE has also been adapted for representing system architectures in conceptual design by Liscouet-Hanke and Jeyaraj to support continued development in further design stages [29], [31]. The thesis is built upon the work conducted by Jeyaraj [29], [31], which serves as the starting point for the research. Several researchers have also investigated the effective use of MBSE [30], [64], [65] in the AGILE 4.0 design framework. The project leverages MBSE technologies and integrates them within an MBSE development system for the modeling, assessment, and optimization of complex systems addressing the entire life cycle. The ambition of model-based system architecting

in the AGILE 4.0 project is to introduce all the development activities of a typical Systems Engineering process and include all the main pillars of the aeronautical supply chain: design, production, certification, and manufacturing [30].

In conclusion, MBSE is well-suited for handling complicated systems and enhancing the development process's efficiency by gathering information and presenting it in customized and relevant views to stakeholders. By using a system model as the primary reference point, there is only one source of accurate information during the development process, eliminating any uncertainty and ensuring consistency in the design process. Using a model-based method allows for thorough tracking and control of requirements on a large scale, which addresses several issues associated with the conventional requirements-based approach [59]. However, the aerospace industry faces challenges in implementing a successful MBSE framework due to the complexity of aircraft systems and the learning curve required to create effective models. Selecting an appropriate MBSE framework is crucial to enable engineers to adhere to standards and manage system complexity and configuration while implementing ARP4754A principles.

2.2.1 Commonly used MBSE tools

MBSE refers to the practice of utilizing system models to facilitate system engineering tasks. A model can take the form of a textual, physical, mathematical, or logical representation of the system [66]. There are two main types of models employed in MBSE: descriptive models and analytical models [52]. Descriptive models capture logical relationships, such as the interactions between system components and functions, as well as the logical and physical architecture of the system. On the other hand, analytical models employ equations, rules, and other direct relationships to represent a system and its characteristics. These models are used in simulations to verify system performance. A system model can be composed of a combination of analytical and descriptive models to represent different aspects of the system. For instance, a system model may be used to specify system requirements descriptively, which can then be mapped onto a simulation model to test system productivity. Moreover, the system model can also be leveraged to identify the safety, reliability, and performance perspectives of the system.

Currently, different approaches and tools are available to implement the MBSE method. Examples of these tools include modeling languages such as SysML (System Modelling Language) and UML (Unified Language Model), and ARCADIA (Architecture Analysis and Design Integrated Approach), as well as representation models like diagram definition and documents [67], [68]. UML, a visual language for complex software systems, uses graphical notations for clear representation [67]. SysML extends UML for system engineering applications and is widely used by the systems engineering community [69]. Various MBSE tools are currently being used to support system engineering activities such as requirements engineering, architecture definition, verification, validation, and model simulation. These tools include Cameo Systems Modeller, Rhapsody, Core, ANSYS SCADE, and the Modelica suite.

Choosing the appropriate MBSE tool depends on the specific application or system that is being evolved. To effectively use an MBSE paradigm, a combination of the modeling language, tool, process, and architecture framework is necessary. A standardized modeling language, such as UML or SysML, is typically used by MBSE solutions, although Capella is an exception. Additionally, a systematic process for developing system models is essential for ensuring transparency and a shared understanding of the system among engineering teams. Unfortunately, many available MBSE solutions do not have an integrated tool and process.

Most tools use a standard modeling language like SysML without providing a modeling methodology. In contrast, ARCADIA/Capella offers a comprehensive solution that facilitates defining a system from start to finish. ARCADIA/Capella is developed and maintained by an open-source consortium with a broad user base, and it has been successfully used by Thales and attracted industry attention due to its flexibility and integrated capabilities. Notably, Capella is extensively used by Bombardier Inc., an industrial partner in the AGILE 4.0 project, making it the tool of choice for this thesis. More information about ARCADIA/Capella is available in the following section.

2.2.2 ARCADIA/Capella

The MBSE ARCADIA (Architecture Analysis & Design Integrated Approach) is a structured modeling framework designed to define and validate the architecture of intricate systems [42].

Thales shift from a supplier to a systems integrator across various domains prompted the development of the ARCADIA methodology. Defining the architecture was anticipated to significantly enhance engineering and system integration effectiveness [70]. A clear understanding of the system at all engineering levels was necessary to improve the V&V process, and early detection of architecture defects and incompatibilities was crucial. All these factors were taken into account in developing the ARCADIA methodology.

The ARCADIA approach uses a standardized modeling process shared among stakeholders to support a structured engineering process. The system model represents the product and connects models at different levels, allowing for collaborative model elaboration across engineering levels [42]. The modeling process captures stakeholders' operational needs and facilitates the system's final integration verification and validation (IV&V). It has been observed that ARCADIA is versatile enough to be used across different engineering disciplines and business units within Thales, and the engineering community has embraced it at different levels. The following text provides an explanation of the modeling process and different levels utilized in the ARCADIA methodology:

The ARCADIA methodology emphasizes utilizing functional need analysis to guide engineering activities. This involves converting requirements into functions and defining functional exchanges, states, and data flows [71]. Although ARCADIA is not tied to a specific tool, the Capella workbench supports it. The Capella workbench offers tools for model development and complexity management, including filters, replicable elements, and copy-paste functionality, among others, to ensure a user-friendly experience.

To facilitate user adoption, the ARCADIA methodology utilizes engineering concepts and language familiar to users, including functions, components, and data, to facilitate user adoption [42]. The ARCADIA methodology consists of several important stages, starting with the identification of user needs and the subsequent development of a solution. This is achieved through operational and system analysis, transforming requirements into clear user needs. In order to meet these needs, logical and physical architectures are created, resulting in solution architectures that fulfill the user's requirements. The ARCADIA methodology allows for the creation of a variety of diagrams based on the system model, which can represent different system views. Figure 2-1 demonstrates the engineering modeling levels that are available in ARCADIA.

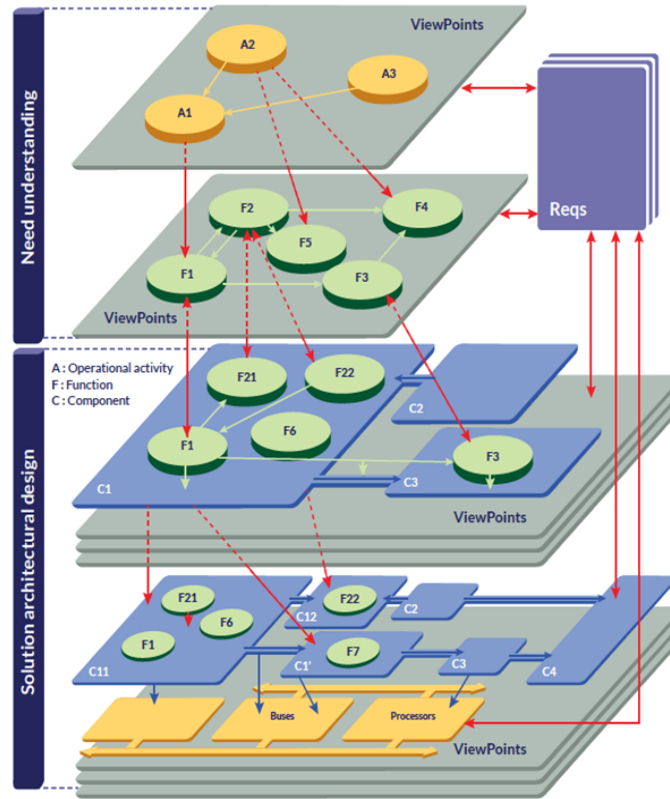


Figure 2- 1: Viewpoints-based approach in ARCADIA, from [68]

The ARCADIA methodology outlines four distinct levels for the process of architecture development, as depicted in Figure 2-1:

1. Operational Analysis Level

In order to define the goals of the system and what the users need to achieve, the ARCADIA methodology uses Operational Analysis (OA). The main aim of this phase is to identify the "actors" who interact with the system and their requirements for the system to operate (for instance, the cabin crew or the flight crew who regulate the cabin temperature). OA identifies use cases or needs through operational capability.

2. System Analysis Level

System Analysis (SA) is a process in which the functions required by the system are determined, and the information flow between functions is developed. SA helps to define what the system must do for the user/actor, such as controlling pressurization. The system begins to take shape at this working level, and requirements are consolidated and formalized. SA identifies and defines the needs or use cases of the system by assessing its capability.

3. Logical Architecture Level

In the Logical Architecture (LA) phase, the focus is on determining how the system operates to meet the required performance. In addition to defining functions, this stage also identifies the components responsible for performing these functions, resulting in a logical architecture. LA presents a logical solution to the needs identified in both SA and OA.

4. Physical Architecture Level

The Physical Architecture (PA) represents the last level in this structure, where it focuses on the physical components of the system and how they will be constructed. Any connections and interactions established between components in the previous levels are maintained, and updates can be made using the transition functionality in Capella [31], [72].

2.2.2.1 Application of Capella in Aircraft system specifications

The ARCADIA methodology has been applied in aircraft systems architecture specification, specifically in the Environmental Control System for integrated modular avionics [60]. It was used to demonstrate the possibility of developing a system architecture specification for aircraft high lift systems using top-down and bottom-up approaches [33] and shows the effective test rig architecture specification for flight control computer [63]. The ARCADIA/Capella has also been adapted to represent the system architecture for effective flight control system and actuation types [29]. It features a two-level modeling approach with generic and technology-specific modeling elements at the system, logical, and physical architecture levels. A catalog of modeling elements representing various actuator technologies for Flight Control System (FCS) is developed to ease the fast generation of system architecture models for design space exploration and subsequent architecture analysis. The steps of architecture representation are illustrated in Figure 2-2.

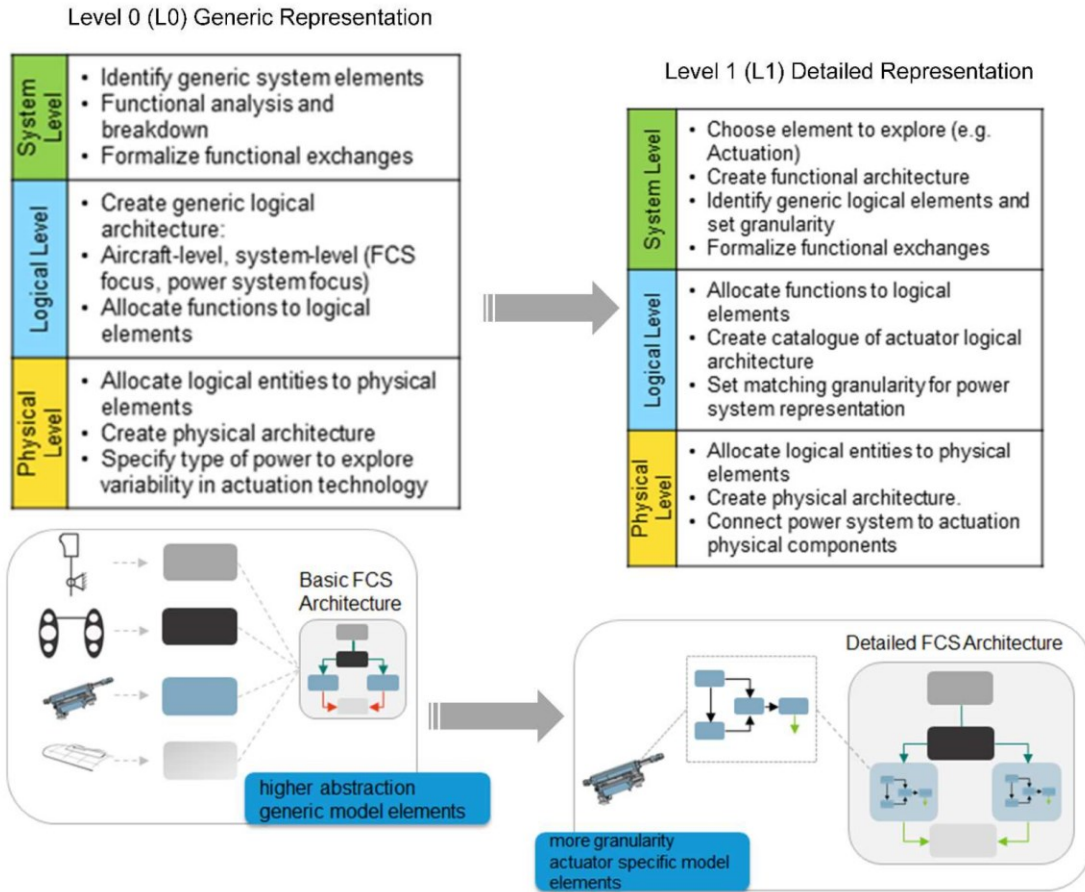


Figure 2- 2: Architecture representations using the multi-level modeling approach, from [25]

These applications are mainly focused on key aircraft systems during the conceptual design phase. The multi-level modeling approach in [31], shown in the figure, is used as a basis to develop the architectural framework presented in this research. The current work aims to create a reusable template of system architecture specification focusing on hybrid-electric aircraft at the conceptual design stage and to facilitate the evaluation of system architectures early in the design stage from different perspectives.

2.2.2.2 Model Enhancement Using Capella Add-ons

Model enhancement refers to the process of making changes or improvements to an existing model to better suit the needs of a particular system architecture. The Capella provides a range of tools and features that enrich essential elements and artifacts, making it easier to develop and refine the

model as the design process progresses. Exploring models from different aspects may result in model enhancement with various properties.

Capella viewpoints are add-ons to the Capella tool, providing extra functionalities for various engineering tasks on system models. These include requirements management, document generation, safety assessment, performance analysis, cost estimation, and time scheduling [73]. Open-source viewpoints are available for certain tasks, enhancing model architectures. Basic viewpoints like mass, price, and performance come with their own property packages, allowing users to augment models with relevant properties. Figure 2-3 shows the engineering activities as Capella add-ons to enhance the architecture models.

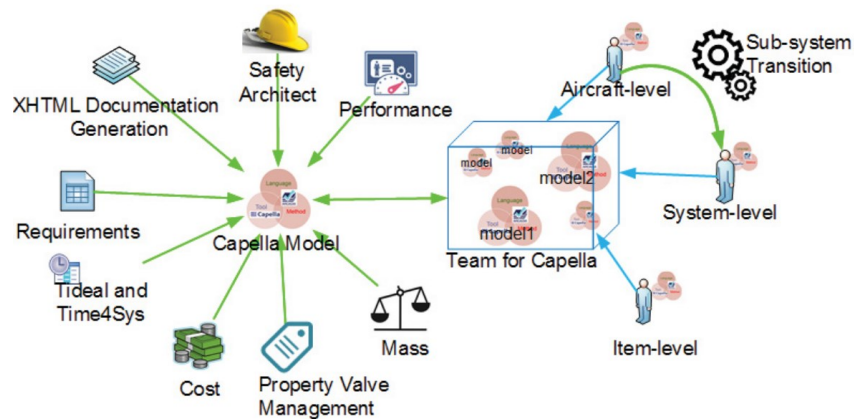


Figure 2- 3: Engineering activities supported by Capella viewpoint extensions, from [69]

Desired Capella viewpoints sometimes need customization or creation from scratch. Capella Studio, an integrated development environment, allows for add-on creation using Java and the Eclipse Modeling Framework (EMF), facilitating standard development that integrates with Capella's architecture. This aids in extending Capella's functionality and integration with other tools. However, Capella Studio's complexity makes it hard for non-programmers to develop simple properties for architecture models, leading to the exploration of a simpler Capella tool in this research, capable of managing properties without any coding. Therefore, In the frame of this work Property Value Management Tool (PVMT) is implemented to enrich models and architectural elements.

PVMT is an add-on to Capella and provides a customized viewpoint that can be used at any level of modeling for any elements, while the other basic viewpoints have limited features and only

cover specific aspects of a model. However, PVMT can be integrated into a model and enhance architecture by giving any custom properties to each element. The PVMT is designed to help system engineers and architects manage the complex relationships between the properties of system components, such as requirements, functions, and physical elements, and the assigned values. The PVMT offers a range of capabilities, including:

1. **Automated property analysis:** The PVMT can automatically analyze the properties and values assigned to system components in a Capella model and provide insights into the relationships and dependencies between them. This can help engineers identify issues and inconsistencies in their models and ensure that they are aligned with the system requirements.
2. **Dynamic filtering and grouping:** It allows users to dynamically filter and group the properties and values of system components based on different criteria, such as requirements, safety reliability, or sizing parameters. This can help engineers quickly identify and isolate issues and focus their analysis efforts on specific areas of the model.
3. **Bulk editing and updating:** It enables engineers to quickly edit and update the properties and values of system components in bulk, which can save time and improve productivity.

Most of the capabilities mentioned above are executable by the PVMT editor, where the property packages are defined and allocated to a modeling layer. In order to create a property package in Capella using the PVMT tool, the property domain should be specified. This refers to the Capella elements and their modeling layer at which the property package applies, such as functions, physical components, or system requirements at SA, LA, and PA modeling layers. They should then be defined by their types, which refer to the property's data type, such as Integer, String, Float, Boolean, or Enumeration. Finally, particular property value rules can be set and applied to the package.

2.2.2.3 Extracting Information from Capella Model

The need for extracting information from Capella arises when designers need to reuse information from system model specifications in another environment and share the model with stakeholders who may not have access to Capella or may not be familiar with this tool. Exporting the model provides a clear and concise representation of the system architecture early in the design stages

that can be easily understood and reviewed by project team members, OEM, suppliers, and other stakeholders. The extracted model from Capella can also be used for documentation, analysis, system behavior simulation, and verification purposes, which are critical for ensuring the system meets the required functional and non-functional requirements. In this thesis, the Capella models are extracted and used for integration within the AGILE 4.0 MDAO workflow. The detailed method of data extraction information from model specifications is addressed in Section 3.3.1.

Capella allows integration with other systems and tools in several manners. Various open-source and commercial extensions are available for Capella to export the model information and data in different formats, as depicted in Figure 2-3. These add-ons provide a way to extract data from Capella models and use it in other applications, making it easier to analyze, report, and share information about the system being developed. XHTML Documentation Generation, an open-source add-on, gives an HTML version of a Capella project to be referenced for other engineering activities during the system architecting process. For model documentation, M2Doc enables the creation of high-quality, customizable, and up-to-date specification files of the models in Microsoft Word.

Recently, a prototype Capella extension, i.e., Python4Capella, has been developed that allows users to interact with their Capella models using Python scripts. With this add-on, users can read and write data from and to their Capella models using Python, which provides a powerful and flexible way to manipulate data. The Python4Capella is currently hosted on Labs for Capella, which is a collaborative platform for the Capella community to share and develop extensions and tools for the Capella tool. It provides an opportunity to explore new ways of integrating models with MDAO workflow and extend the functionality of the Capella using Python. However, the Python4Capella is still a prototype and may not be fully functional or may contain bugs or limitations.

Above all the possible extraction methods, there is a simpler and more practical solution to export model specification data for further applications. Capella export feature enables system information and component properties to be incorporated into the Comma-Separated Values (CSV) format that can be opened in other applications such as spreadsheets or databases. The CSV export format includes a set of rules for formatting data, where each row represents an instance of a particular element or object in the Capella model, and each column represents an attribute or

property of that element. For example, a CSV export of a Capella model might include a row for each requirement in the model, with columns for the requirement ID, name, description, status, and other relevant attributes. Similarly, a CSV export of a Capella model might include a row for each component in the model, with columns for the component ID, name, type, interfaces, and other relevant attributes. This feature in Capella is used during this research for sharing and documenting information from models created within this framework.

To illustrate system architecture models, there is an image export option in Capella that enables users to save and share the model at any stage of the model construction in JPG, PNG, SVG, BMP, and GIF formats. All the system architecture presented in the Appendix is extracted in a JPEG file by using this feature.

In summary, MBSE has emerged as a powerful method for designing complex systems. ARCADIA stands as a transformative approach within MBSE, providing a standardized and structured pathway for defining and validating intricate system architectures. It allows for the creation of a comprehensive system model that can be used to visualize and evaluate the system's architecture. Using these models, system architects can identify potential design flaws and trade-offs before implementation, saving time and costs. The following sections discuss how other researchers utilize MBSE in system architecture evaluation and present a deeper exploration of the MBSE bridge to MDAO and MBSA environments.

2.3 Systems Architecting Considerations in MDAO Workflow

MDAO is an engineering design methodology that involves the simultaneous optimization of multiple disciplines or subsystems to improve the overall performance of a complex system. MDAO requires a detailed understanding of the system's architecture specifications which can be achieved through MBSE. MBSE enables the creation of system models to perform MDAO. According to [8], MDAO can be integrated with MBSE by creating models that represent the system's subsystems and disciplines. The models can be used to perform system-level optimization, which involves optimizing the system's architecture and requirements to improve overall system performance.

During the conceptual design phase, system architecture activities should be conducted to ensure that the ultimately chosen architecture, which is further refined and applied to the aircraft, meets the requirements most effectively. MBSE is incorporated into the design process to create a specific system architecture during the detailed design phase [74]. However, the adaptability of MBSE to conceptual design activities is relatively limited due to the complexities associated with managing various potential architectures and variants within a layered, complex specification model [29], [33].

The current process in the industry lacks MBSE during the exploration phase, limiting the evaluation of a few system architectures. The specifications of architectures are manually transferred to MDAO workflow at the system level for evaluation. The MBSE specification phase is typically disconnected and used after this initial phase. While some academic approaches enable the evaluation of many system architectures, these approaches are not widely adopted in the industry [25]. Figure 2–4 visualizes the current situation (“as-is processed”) and the envisaged future process (“to-be process”), addressing the drawbacks of each step of design space exploration.

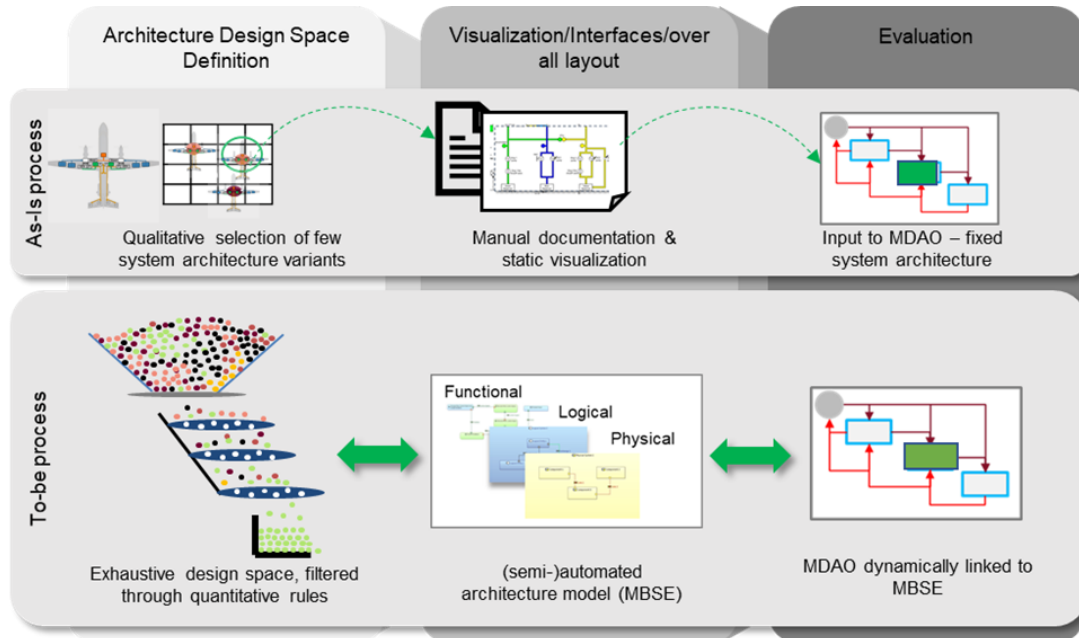


Figure 2- 4: Stages of the system architecting process for design space exploration

To evaluate the system architecture in conceptual design, sizing and analysis tools are utilized by the designer. These tools can be incorporated into a system-level workflow within an MDAO

environment at the aircraft level. For this reason, a strategy is needed to ensure that the system-level workflow obtains information from the system architecture for both configuration and execution. The system architecture is then assessed to determine critical performance metrics, including mass increments, drag, fuel, and other aircraft-level parameters, as stated in [44], [37]. The evaluation workflow can be standardized and easily replicated with minor changes for conventional and innovative system architectures [44]–[46]. However, configuring the workflow may be time-consuming for unconventional and innovative architectures, as models need to be identified, sizing and performance routines synthesized, and a connection to aircraft-level metrics established.

The connection between MBSE and MDAO is important to access the information about the system architecture contained within the model-based specification, which is then used to configure the MDAO workflow. The model-based specification also provides tool inputs that are assigned to model elements for different tools in the system-level workflow, making it a crucial part of the updated system architecting process. However, challenges remain in enabling this capability, such as identifying and categorizing the transfer of information from the MBSE specification to the MDAO environment [21], [22]. The inadequate exchange of information between system architecting and MDAO frameworks can hinder integration. This raises the question of how architecture description artifacts can be mapped to workflow elements in an MDAO specification. Significant efforts have been made in developing MDAO frameworks with reconfiguration capabilities that support collaborative MDAO processes, but the integration of MDAO methods specifically to support system architecting activities is still lacking.

There are several challenges based on a literature review by Chaudemar et al.[75] on the concurrent application of MBSE and MDAO early in the design stage. Formulating the MDAO problem consumes significant time, even though the required information is readily accessible in the MBSE system model. Moreover, The integration of MDAO solutions into the MBSE system model is not discovered sufficiently due to the MBSE tools and language limitations that require manual effort. Aiello et al.[76] Propose a method to extract system requirements from the MBSE model and enhance them through MDAO in five steps. The method is tested for sizing a drone battery using papyrus as an MBSE tool and OpenMDAO. However, their proposal does not address the reuse of MBSE models.

Ciampa et al. [77] use an architectural framework to link the MBSE and MDAO workflow. This method introduces a 'development system' that assists in the evolution of the 'system of interest' (SoI) at various stages, using MBSE to quicken the creation of MDAO systems. This involves a formal description of the MDAO system's structure and behavior through an MDAO system architecture. This architecture uses a framework that is a combination of terminology and viewpoints for architectures, requirements, and lifecycle. In this context, system architecting, specification, and identification are formalized through MBSE and linked to MDAO. However, the reuse of the system model and data is not described, and the execution of MDAO for sizing and parametrizing the system of interest is not addressed.

Bussemaker et al. [8] offer MBSE Architecting to MDAO bridging approach by using function-based architecture modeling and collaborative workflows that leverage disciplinary expertise. The architecting step builds on the requirements specification step, and therefore a large part of the requirements validation can also be taken up by the architecting activity through the specification of architecture properties and formulation of the architecture design problem. The architecting approach focuses on modeling the function-based architecture design space for architecture optimization. The MDAO approach implements the collaborative MDAO principle to leverage disciplinary expertise and implement cross-organizational MDAO workflows.

Habermehl et al.[78] propose a formalization of the MDAO workflow by three modeling approaches: workflow architecture, internal behavior, and executable behavior. The main purposes of the formalization are the resolution of the MDAO black box behavior and the provision of a means to parameterize the system architecture with an optimal parameter set. The paper aims to provide a novel approach to link MDAO in MBSE system models, taking advantage of existing models and thus reducing the initial implementation effort of MDAO. The workflow's structure and behavior are formally outlined using SysML for an electric coolant pump as a use case

Moreover, The integration of safety and certification considerations into the system model can be accomplished using MBSE and MDAO, where the model can provide this information as constraints to the MDAO workflow [79], [80]. However, evaluating a large design space through the architecture evaluation process is expensive in terms of computation cost, requiring pre-filtering of viable designs.

In recent times, there has been a focus on increasing automation in the implementation and execution of MDAO [21], [81]. To make the formulation phase of MDAO more efficient, a centralized MDAO schema and MDAO definition environments have been developed. These environments consolidate tool repositories and allow for the structuring and visualization of MDAO workflows [81]. By using these schemas, MDAO specifications can be designed and executed automatically. In general, MDAO specifications, implementation, and execution have become more advanced and automated, and they have the potential to be easily reconfigured. However, many organizations still follow the traditional approach of using Process Integration and Design Optimization (PIDO) tools, which rely on manual effort for reconfiguration.

Ultimately, the MDAO specification needs to be executed in software, necessitating a robust data framework for storing, exchanging, and processing information about MDAO workflow configuration. The Common MDAO Workflow Schema (CMDOWS), as defined by [81], coupled with the Common Parametric Aircraft Schema (CPACS), offers a mechanism for capturing information related to the MDAO workflow in a neutral data format. This information can then be processed by software and visualization frameworks for execution and user interaction [82].

In summary, despite recent development, there is still a lack of connection or integration between linking MBSE system architecting and MDAO, which is challenging as they have different characteristics in development, formalization, fidelity, and reconfiguration that require a deeper exploration and effort. System architecting is a flexible and manual process that involves many iterations and changes, while MDAO is automated and static. This integration can be achieved by identifying the necessary information that should be shared and examining the current structure of MDAO data through analysis of exchange standards and framework.

2.4 Model-based safety assessment (MBSA)

Model-based safety assessment (MBSA) is a method that uses system models to evaluate system safety. This approach differs from traditional safety analysis methods, relying on models rather than manual analysis of system components. The connection between system modeling and safety assessment has been acknowledged as an essential aspect of model-based safety assessment processes. MBSA can be used to identify potential hazards and assess the safety of a system design,

allowing designers to make informed decisions about improving their systems' safety and reliability [7]. MBSA utilizes formal modeling techniques to aid in safety assessment tasks. These models can be classified as either standalone or extended models [83], [84]. Standalone models are designed to represent safety-related information by capturing the relationship between the constituent elements of a system. On the other hand, extended models are safety-specific abstractions of the parent system model, focusing on safety-related aspects. Current approaches to MBSA involve using extended models that are developed in specialized environments, such as MATLAB/Simulink.

The industry is investigating progress toward the MBSA approach [7], [83], [85]–[87]. Recent developments in MBSA applications mainly focus on using a model-based approach to perform safety analyses such as Failure Mode and Effects Analysis (FMEA) and Fault Tree Analysis (FTA). Gradel et al. [7] have demonstrated the use of MBSA in conceptual design by illustrating the development and evaluation of fault trees using a model-based definition of failure events that includes safety requirements obtained from FMEA. The proposed scheme extends the classical FMEA scheme with fault propagation behavior to enable a more comprehensive analysis of potential hazards for unconventional aircraft systems, while Bruno et al. [88] have proposed a Reliability, Accessibility, Maintainability, and Safety Framework for conventional as well as more electric system architectures, which includes a model-based FMEA.

A proposed link between MBSE and MBSA by Sango et al. [89] is performed using ARCADIA/Capella and SafetyArchitect tool developed by ALL4TEC. It leverages the Safety viewpoint in Capella and the import legacy in Safety Architect to conduct classical safety analyses such as FTA. However, the MBSA tool used for the study is not open-source; therefore, the accessibility, compatibility with other tools, and customization capabilities are limited.

Maitrehenry et al. [90] propose a new method for identifying failure scenarios in aircraft systems that is operations and flight-phase-oriented, and that increases confidence in the analysis of failure combinations. The study also highlights the importance of safety in the aviation industry and how formal model-based FHA can be used to verify that proposed aircraft architectures fulfill their safety requirements. Villhauer et al. [91] illustrate a method for performing FHA in a SysML environment by presenting an aircraft pitch controller model as a test case.

Recently, there has been advancement in integrating FHA into the system architecting process. Jimeno et al. [92], [49] offer a safety analysis comprised of two methods. The first method supports semi-automating the FHA process, using the functional view as input. The FHA results are then used to update the safety objectives in the requirements view. The second method automatically generates Fault Trees from the Logical view, which can be evaluated both qualitatively and quantitatively. The contribution to the probability of failure of their components can be ranked through importance measures, helping the architect to focus on the most problematic parts of the architecture. An MBSE-based FHA approach investigated by Jiang et al. [93] presents the implementation of an FHA using SysML for the landing gear braking system test case from the ARP4761 example.

Overall, several researchers have investigated and employed MBSA for different safety analyses, such as FHA, FTA, and FMEA, using model-based approaches. MBSE enables the creation of system models that can be used to represent the system's architectures and behavior. Integration of safety analysis within the MBSE framework is important as it allows for the development of safer and more realistic system architectures, especially for unconventional and innovative technologies where new architectures have to be designed and system safety has to be taken into account. By leveraging the functional structure of the architecture, integral processes such as safety analysis can be performed, and artifacts such as reusable modeling elements can be developed, resulting in greater efficiency in system integration and ultimately reducing development cost and time. MBSA within a model-based environment also allows for the use of advanced safety analysis techniques, such as FHA, that can be integrated with the system architecture model to identify and evaluate safety concerns early in the design process.

A systematic process is necessary for integrating the FHA with MBSE, which includes determining the appropriate level of elaboration, creating specific diagrams and viewpoints, and customizing them to assist the safety analyst in examining functional chains, hierarchies, and allocations within an MBSE environment. Additionally, it is necessary to consider enhancing the model-based system architecture specification artifacts with safety-specific information to support downstream MBSA activities, such as fault tree analysis. Lastly, there is a need to demonstrate the integrated FHA process in developing new aircraft system architectures to enhance the overall aircraft development process.

2.5 Summary and Gap Analysis

MBSE is promising for managing complex systems and enhancing the efficiency of development processes. However, the aerospace industry faces challenges in adopting an effective MBSE. Therefore, it is crucial to select a suitable MBSE framework to manage the complexity of unconventional aircraft and be adaptive to different system configurations. In this research, ARCADIA/Capella is selected as a tool to generate generic templates of architectures for hybrid-electric propulsion systems.

The integration of MBSE to MDAO still lacks a direct link using a model-based system architecture specification in Capella that contains all the necessary system specification information. This represents an area of unexplored potential in which further investigation and development in this thesis could lead to more efficient and comprehensive evaluation and optimization of the design models.

Moreover, there is a need for a methodical procedure to integrate FHA within the MBSE framework. The augmentation of model-based system architecture specification with safety-centric information needs to be considered. to facilitate downstream MBSA activities early in design stages. This can be achieved in this thesis by introducing and performing a model-based FHA for system architectures in Capella as an initial step toward comprehensive MBSA.

3 Methodology

This chapter provides an overview of a multi-level approach within the MBSE framework to demonstrate the architectural development of novel aircraft systems such as hybrid-electric aircraft. Then, it presents the integration of model-based system architectures in Capella to the MDAO workflow to advance the system architecting evaluation phase. It is followed by a proposal on how to enrich the MBSE specification to support FHA as a step towards enabling MBSA.

Figure 3-1 shows the activities breakdown required for architecture representation with different levels of granularity in Capella and the connection to other environments involved in the system architecting process. It features a multi-level modeling approach that establishes generic architectures and variants of the aircraft system elements that can be reused to build advanced system architectures.

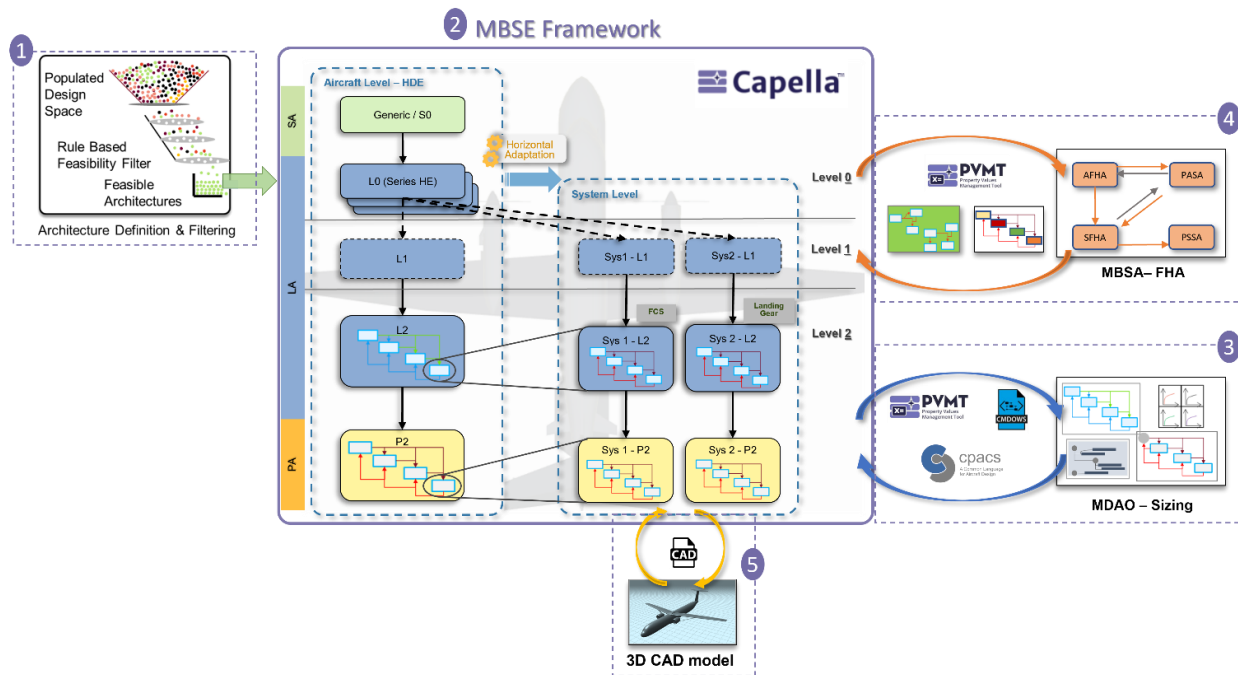


Figure 3- 1: Overview of the methodology in Capella and connections to other analysis environments

The five sections numbered in the overview of methodology are:

1. System architecture and design space definition, and rule-based filtering (this part is not in the of scope of this thesis but covered within the AGILE4.0 project)

2. MBSE multi-level framework in Capella
3. MBSE to MDAO sizing tool environment in collaboration with the AGILE 4.0 project, this is also covered in the conference paper [94]
4. MBSE to MBSA bridge supporting FHA, this is also addressed in the conference paper [95]
5. MBSE to 3D modeling CAD design as future work (this part is not in the of scope of this thesis but was planned to be addressed within the AGILE4.0 project)

The focus of this thesis is mainly on phases 2, 3, and 4, which are discussed in detail in this chapter. However, the methodology was built to also easily connect with activities in points 1 and 5 in Figure 3-1.

3.1 Multi-level System Specification

The proposed multi-level modeling approach is a technique that helps manage complexity and create model-based architecture designs for unconventional aircraft, which feature a high-level of variability in early design phases [31]. The multi-level approach needs to be defined in relationship to the modeling levels defined in a specific MBSE solution; here, the ARCADIA/Capella solution is used.

The multi-level approach in the MBSE framework involves building models at different levels of detail, called granularity. The granularity of system artifacts increases through a top-down approach, which begins with defining high-level components and system requirements and then moves to lower-level components and detailed artifacts. This process ultimately results in a complete physical architecture for the aircraft.

The multi-level approach is particularly useful for iterative and concurrent system development processes. It provides an effective conceptual model for the system development process to develop a specific system implementation from knowledge of an intended aircraft function. The approach involves creating system functions and allocating them to logical components, which are then allocated to physical components to comprise a complete physical architecture of an aircraft configuration.

The proposed method allows the creation of basic architecture diagrams and generic configuration variants for unconventional aircraft systems and subsystems. The generic model then evolves into a more detailed architecture, resulting in a complete architecture for the aircraft.

3.1.1 Model Development

Capella's modeling process follows the ARCADIA method's disciplines, depicted in Figure 2-1 in Chapter 2, and it is formalized to support the multi-level modeling approach at conceptual design for unconventional systems in aircraft. The proposed top-down methodology maps the MBSE steps to ARP4754A practices [23] for aircraft and system development while it follows the ARCADIA's modeling convention [42].

The system architecture design space is filtered for feasibility based on predefined rules. This results in a reduced set of architectures that are modeled within an MBSE environment. Then, a few feasible architectures are selected as input for this research, and the primary requirements for developing the architectures are considered. Figure 3-2 shows architecture models catalog in various levels of detail across Capella modeling layers. The purpose of this approach is to build reusable system architectures in an MBSE framework. As the conceptual development process is performed for the aircraft/system-level functions, the proposed modeling framework is outlined in a way to cover both levels in models with different granularity.

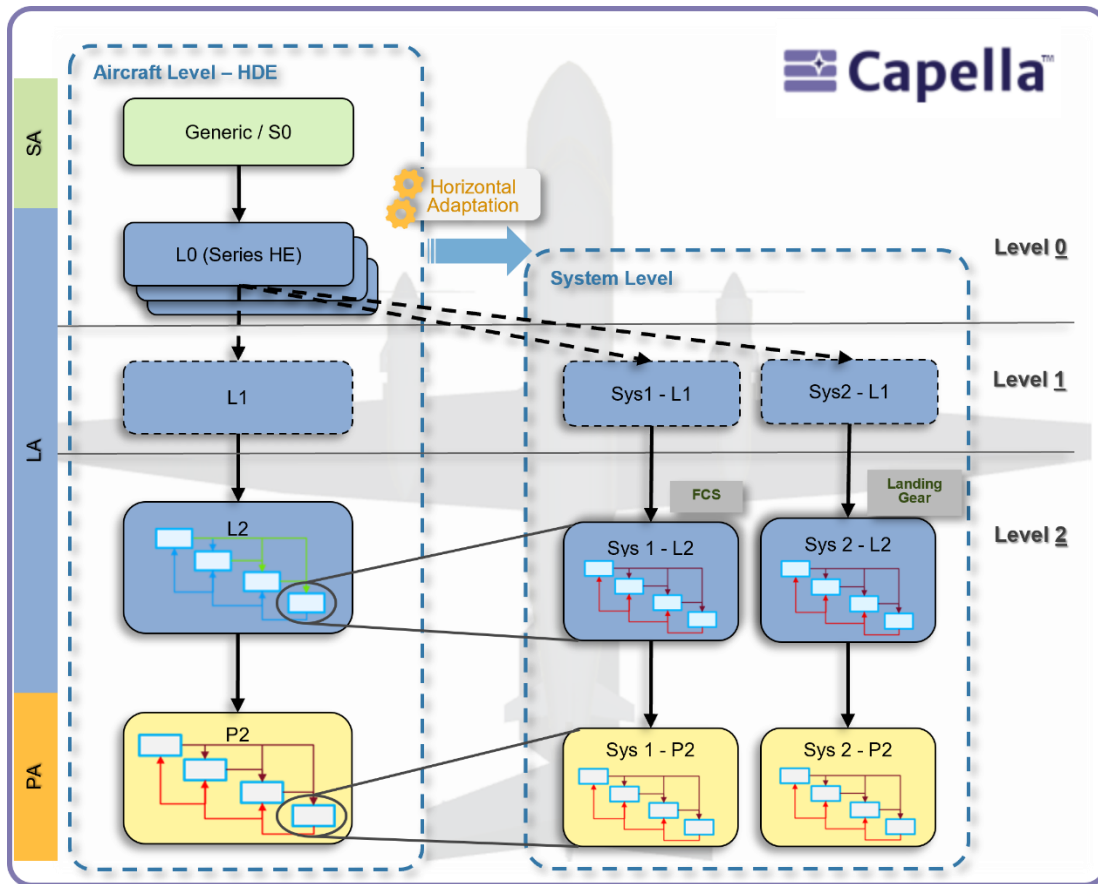


Figure 3- 2: Multi-level modeling approach and levels of detail within the MBSE framework

The multi-level framework is implemented in Capella, which enables handling architecture variants by reusable model representation artifacts. Architecture specifications used in this method contained reusable elements, including functions and logical and physical components. Model artifacts are replicable and reusable in several models at different modeling stages throughout an elemental catalog in Capella, which is described in detail in Section 3.1.4.

According to Figure 3-2, the multi-level modeling framework covers two primary engineering levels, as discussed in Chapter 1, aircraft-level and system-level:

1. Aircraft-level consists of the highest level of aircraft functions and systems. This level only covers the major systems and their interfaces without any subsystems. For instance, in the case of hybrid electric aircraft architecture, aircraft-level systems are the electrical system, the propulsion system, and FCS.
2. System-level functions and subsystems are derived from higher levels that follow a top-down approach. The system level focuses on how the detailed functions of a specific system are

allocated to the subsystems and shows the interfaces between subsystems and other systems of aircraft.

At each level, functions and logical and physical components can be defined to build up architectures. However, when integrating architectures is important to ensure consistency between the aircraft-level and system-level functions. The model variants management is discussed in detail in Section 3.1.3.

In a Capella-based approach, the different modeling layers represent different analyses and are typically performed by different people. It is a way of separating the different analyses while maintaining consistency across the levels. This method is best suited in cases where the system architecture is not well defined or if the system architecture needs to be developed from scratch.

Developing a system in Capella usually starts at the Operational Analysis level (OA), where requirements, entities, and actors to complete a mission are defined. The operational analysis capture market and customer needs; however, this research focuses on system architecture development, and the operational analysis is assumed completed.

The work of the system architect, as assumed in this thesis, starts with realizing aircraft/system-level functions, as the requirements and the system of interest are known. Instead, the multi-level specification can be built using system Analysis (SA). The system architecture design space starts here. At this layer of modeling in Capella where the system engineer defines the system of interest and the aircraft-level functions to build a generic architecture, which can later be tailored for a range of aircraft requirements and support a variety of aircraft configurations and architectures. For example, in the hybrid-electric propulsion case study, the scope of the system(s) and components are described in design space definitions.

The next modeling level in Capella, the Logical Architecture (LA), is used to determine logical components and their relationship in the logical architecture of the system(s). Depending on the aircraft configurations, logical architectures in different levels of detail can be derived from the generic architectures. At this modeling level, logical functions at the aircraft and system levels are allocated to so-called logical components. The logical component provides a logical solution or behavior for the deployed functions.

Finally, the LA is passed down to Physical Architecture (PA), where physical components (hardware and software) and the various types of interfaces are associated with logical components and exchanges. The logical components and functions are at the maximum level of granularity.

This methodology allows the creation of an accurate representation of the system interfaces and exchanges at logical and physical levels during the conceptual design, which can then be expanded at preliminary design and provided to suppliers and manufacturers for further development. It offers architecture representation for various models at the LA and PA levels for different systems configurations in Capella. Architecture models for case studies of flight control system hydraulic and fuel systems are presented in Chapter 4.

3.1.2 Levels of Detail in the Modeling Approach

As explained in the previous section, the modeling methodology needs to cover aircraft and system levels. The process of developing models in Figure 3-2 is organized into three levels of detail, each adding certain information and value to the model(s), which are briefly described in Table 3-1.

Table 3- 1: Levels of detail specification in the Capella modeling methodology

Detail Level	Description
Level 0	A generic model for aircraft configuration, significantly on propulsion system – including top-level functions and generic interface elaboration
Level 1	Intermediate level definition of aircraft characterization, Including aircraft-level redundancy and key technological choices – number of engines, number of hydraulic systems, and level of electrification/hybridization
Level 2	Includes sufficient elaboration of subsystems or logical components to show allocation

- Detail level 0 is characterized by the highest level of abstraction, and it covers the top-level aircraft functions for a generic aircraft configuration. As the SA model architecture is generic, it can fit into level 0, and the model is called Generic S0.

- Detail level 1 is a transitional level between aircraft-level elements and system-level functions in models. Therefore, models are designed to be set for aircraft-level and system-level functions. They include logical component redundancy related to technological choices. For example, in the hybrid-electric case study, the number of engines needs to be defined in the early design stage. LA models are available in granularity levels 0, 1, and 2.
- Detail level 2 includes component redundancies at both the aircraft-level and system-level. PA and LA models at this level of detail illustrate the elaboration of the (sub)systems for a particular configuration and the number of system components and items.

Here are examples of aircraft-level models at the LA modeling layer, from Figure 3-3 to Figure 3-5, illustrating the model specifications of a hybrid-electric aircraft in the three levels of detail. In the process of developing models within the Capella framework, substantial emphasis is laid on the System Architecture Breakdown [SAB], the Logical Architecture Breakdown [LAB], and the Physical Architecture Breakdown [PAB] diagrams. These diagrams are instrumental as they provide a comprehensive overview of the system, components, allocations, and exchanges. Other types of Capella diagrams related to the frame of this work are discussed in section 3.5. System interfaces and functional exchanges are highlighted to show their significance in architecture. The HE configurations and their differences are discussed in Chapter 4, Section 4.1.

The architectures in the LA layer are also categorized by their level of detail. The logical model detail level 0 (L0) is a generic model similar to S0, only with logical components where the top-level functions are allocated to the systems. At this level of modeling, high-level architecture variants of the L0 model can be created; they are distinguished by their functional exchanges and data flow in their architectures since most architectural elements are common.

In this thesis, the three configurations of the HE propulsion system, namely the series, parallel, and partial series-parallel architectures (see Section 4.1.1 for more details), are created. Figure 3-3 shows the L0 model for the parallel configuration of the HE propulsion system to show the modeling granularity. Note that the System of Interest (SOI) in all the architectures is “Aircraft”. However, the SOI box is removed only for representation reasons in these models, and the focus is on illustrating its (sub)systems.

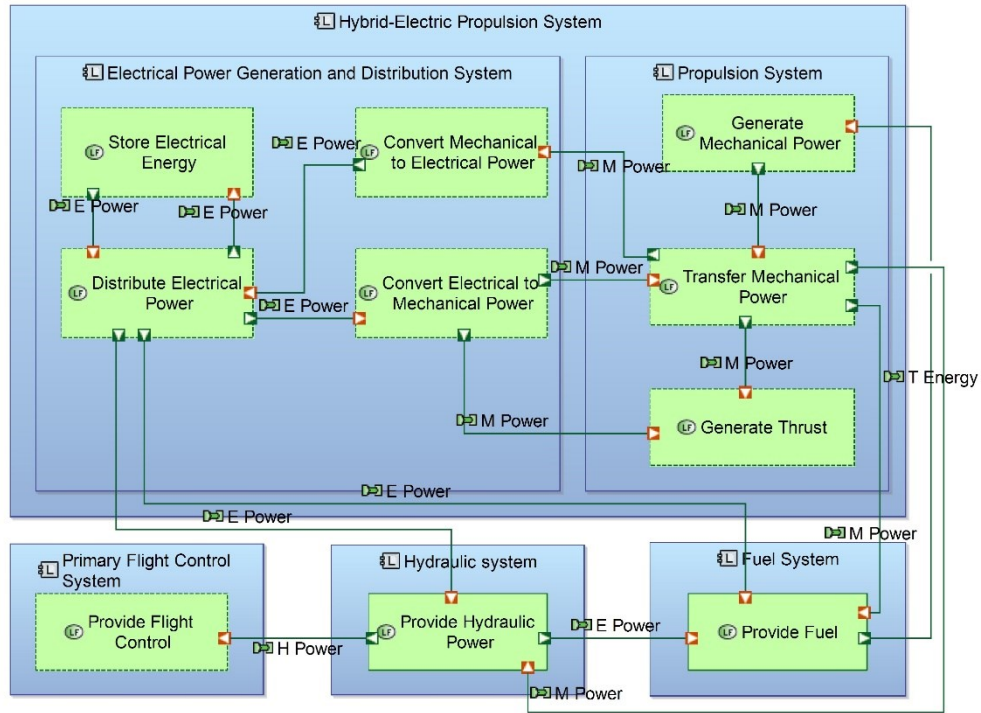


Figure 3- 3: L0 model aircraft-level model of hybrid-electric aircraft partial series/parallel configuration at the LA layer with detail level 0

Figure 3-3 shows the hybrid-electric propulsion system in more detail than the interfacing system, as this part is the system of interest in this study. However, examples of major interfacing systems, such as the fuel system, the hydraulic system, and the primary flight control system, are also shown, as they will be further investigated in the case studies (Chapter 4), and they present various related subsystem levels.

Figure 3-4 shows logical model with the detail level 1 (L1) for the parallel configuration of the HE aircraft. In this model, some top-level functions of the L0 model are broken down into more specific functions, and logical components are allocated to the systems. For the HE case study, model L1 includes the number of engines and the propeller types, such as the main propeller and the small propeller as support. These technological decisions need to be addressed early in the conceptual design stage. Note that the level of granularity of the interacting systems (Primary Flight Control System, Hydraulic System, and Fuel System) remains unchanged.

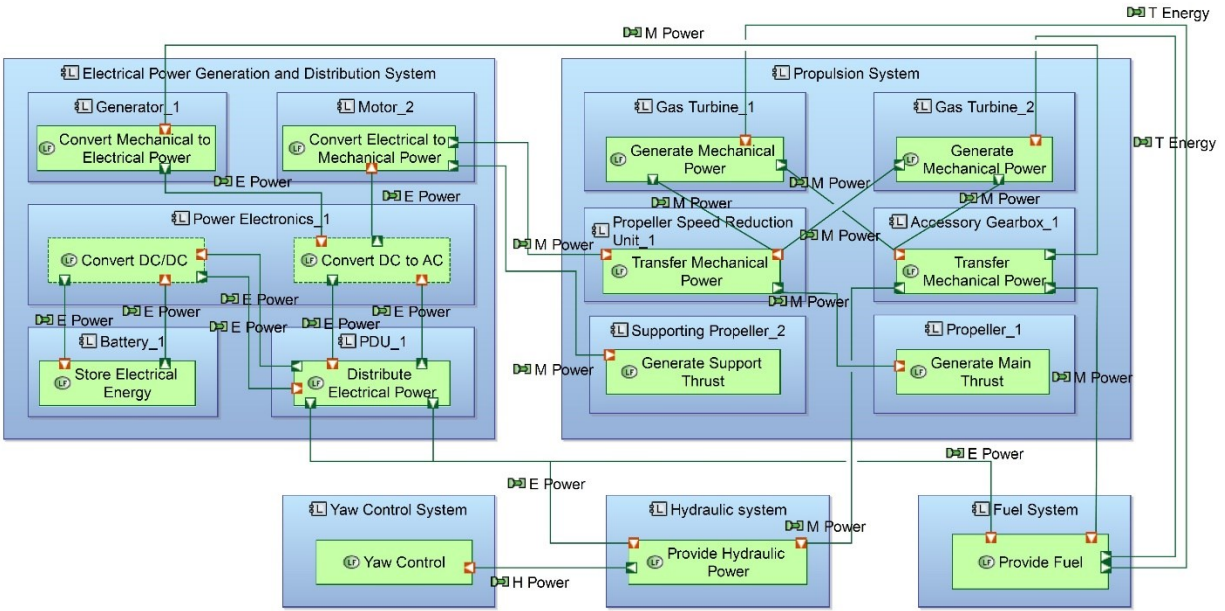


Figure 3- 4: L1 model aircraft-level model of hybrid-electric aircraft partial series/parallel configuration at the LA layer with detail level 1

The logical model level 2 contains all the component redundancies in the aircraft, and the architecture is too detailed to demonstrate. Therefore, Figure 3-5 shows the propulsion and electrical systems cropped from the model at logical architecture detailed level 2 (L2) for the HE case study. Models at this level, either at LA or PA, contain allocation of system requirements to items. The complete models at these levels are presented in Chapter 4 in detail.

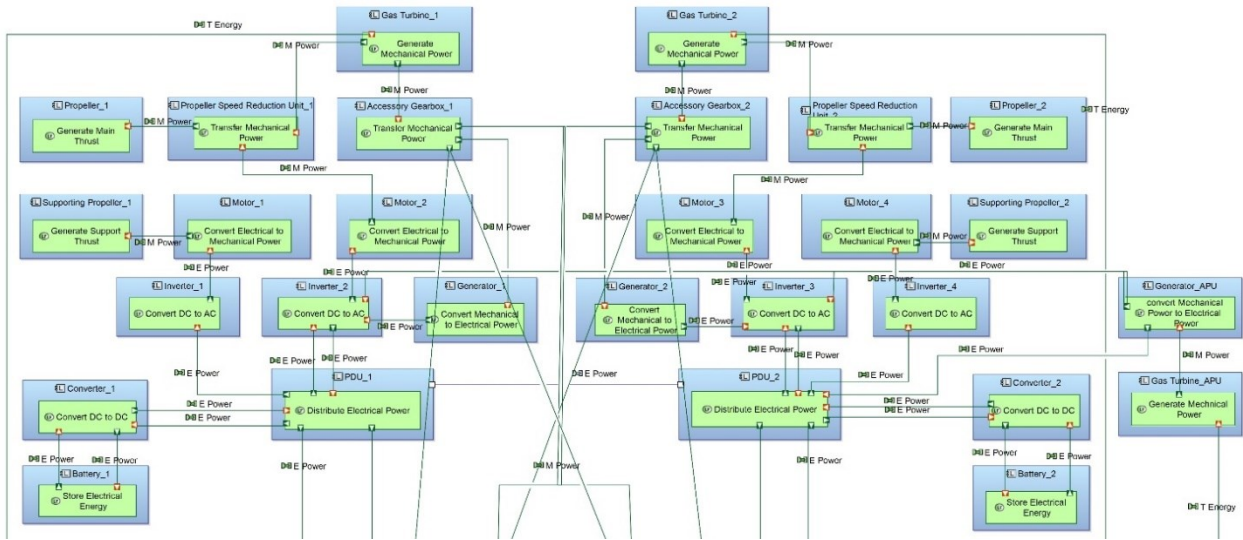


Figure 3- 5: L2 model aircraft-level model of hybrid-electric aircraft partial series/parallel configuration at the LA layer with detail level 2 for propulsion and electrical systems

This figure is extracted from a larger L2 model depicted in Figure 4-9 in Chapter 4.

3.1.3 Model Variants Adaptation & Management

The described strategy for system development and building models is performed across Capella modeling layers at aircraft and system levels. At the LA and PA modeling layers in Capella, several models can be built for the (sub)system(s). In this modeling approach, a main model with aircraft-level functions can be propagated into several models with system-level functions. The derived models are designed to capture the elaborations of logical and physical components of (sub-)systems, along with their interfaces with other systems. At the LA layer, aircraft-level model propagation is started at logical model detail level 1(L1). Then the derivative models are referred to as System-Logical with level 1 detailed (Sys-L1). The Sys-L1 is a centralized and expanded model of a logical component in the L1 model. The same rule is applied for the logical model detail level 2 (L2). Therefore, for any specific system in aircraft, Sys-L2 can be built. For L2 and Sys-L2, physical architectures are defined as P2 and Sys-P2, respectively. In order to manage the model variants for better accessibility and efficient traceability of the system architectures, two approaches are available for Capella to investigate: the so-called *Horizontal Adaptation* and *Vertical Transition* [42].

In the present work, model variants are integrated within the modeling layers by horizontal adaptation to avoid inconsistency through the aircraft to system levels transition and model development. This method provides insights at the system level without losing sight of the overall picture at the aircraft level. Besides, this type of adaptation maintains the architectural unity of different aircraft systems. Figure 3-6 demonstrates the horizontal adaptation method applied at the LA level, transitioning from the L1 model to the Sys-L1 models. For example, the Primary Flight Control System in the L1 model is decomposed to one of its sub-systems, Yaw Control, in the Sys1-L1 model. Similarly, the Hydraulic System in the L1 model is decomposed into detailed components in the Sys2-L1 model, containing system-level functions.

Horizontal adaptation involves using a generic architecture for system specifications while defining the behavior of subsystems through encapsulation. This means that the functions of the subsystems are incorporated into the parent functions of the main components without focusing

on the subcomponents themselves [72]. The functions are encapsulated in the Sys-L1 and sys-L2 models.

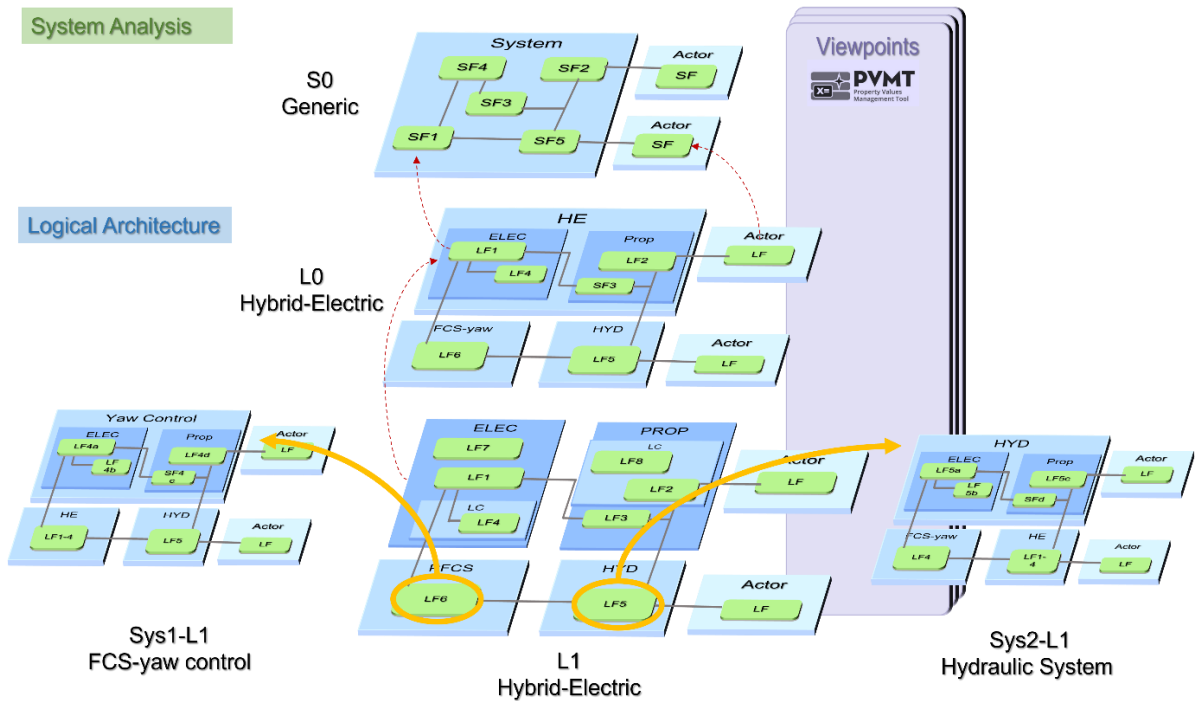


Figure 3- 6: Horizontal adaptation for aircraft to system-level transition

It is important to note that each subsystem requires its own separate model to implement the MBSE method for aircraft system development effectively. This is due to the limitation of Capella to allow “zooming” into sub-systems within the same model. However, the horizontal adaptation method is not suitable for detailed subsystem specifications; it is only utilized when the detail of subcomponents or the behavior of the subsystem is not required [60]. In this work, the focus is not on detailed specifications of subsystems. Instead, it focuses on the system specifications and their interfaces at aircraft and system levels, and it attempts to depict the transition from aircraft level to system level models without losing the unity of all the systems.

Another way to manage the model variant is “vertical transition,” which involves defining the detailed architecture of a system's components by realizing their subcomponents. Capella subsystem-transition add-on is required to perform the vertical transition of a logical component to lower levels of detail in the system architecture (SA). In this method, main sibling logical components interacting directly with the selected subsystem become system actors. This would

result in a clean and efficient model. However, the vertical transition method is more effective when a particular subsystem or logical component is specifications are required, and the high-level model is finalized; this is rarely the case in the conceptual design iterations. As the vertical transition approach only focuses on the sibling component or actor, the overview of the entire system might be lost. In other words, it enables a more targeted and efficient approach to modeling and designing a subsystem. Therefore, the vertical transition is not suitable to be implemented in the proposed framework. Instead, horizontal adaptation has been chosen to manage the models across the framework.

3.1.4 Architectural Element Catalog & Reusability

The architecture specification uses a catalog of architectural artifacts, including logical and physical components and functions that have been defined in Capella with different granularity levels. These artifacts are appropriate to be deployed frequently in architectural representations at the conceptual design stage. The advantage of defining these basic architectural artifacts and elements at an early stage is that once the system architecture has been selected, the basic elements that are already contained within the model-based specification can be easily expanded and refined as the design process progresses. This saves time and effort in the later stages of design by providing a foundation of well-defined elements to build upon. Most of the logical and physical architecture elements are made reusable using the Replicable Elements Collection (REC) in Capella. The REC allows architectural elements to be reused across different models, configurations, and contexts. This feature enhances the reusability of architectural artifacts and helps to maintain the relationships between exchanges, logical and physical components, and functions. The REC elements are instantiated as RPL (Replicas) within a specific configuration or model, allowing them to be used in multiple models. By utilizing the REC feature, Capella enables the efficient reuse of modeling elements and ensures that the relationships between the different components and functions are kept intact [96].

In this context, referring to Figure 3-5, the logical component “Motor” and its function “Convert Electrical to Mechanical Power” are made reusable by REC and saved into the Capella library. These elements are replicated (RPL) in Figure 3-5 (L2) and other models as required. Therefore, the redundancy of the components in Capella models within this method is addressed with

REC/RPL feature while constructing the complete system architecture representations. Having reusable elements and artifacts speeds up the process of building architecture representations and also results in the creation of modeling artifacts that can be developed and enhanced later.

3.1.5 Model Nomenclature Consistency

Another aim of this work is to create reusable templates of aircraft and system within the MBSE framework. This helps implement the system architecture development of ARP4754A with predefined and reusable architectural elements. Through the architecture representations at different granularity levels from SA to PA layers in Capella, each model contains particular information and components and functions at a specific level (i.e., aircraft, system, or item levels). However, it is important to keep the consistency of elements and artifact naming while using the architectural representation templates for any system modeling practices. To this end, a list of Capella model elements, including functions, logical components, logical behaviors, and physical implementation, is created for each system in aircraft. A part of this nomenclature is depicted in Table 3-2 as an example of elements used in the propulsion system. The complete list, including propulsion, electrical, hydraulic, fuel, and yaw control systems, can be found in the Appendix. All the functions at the system, logical and physical levels which referred as green boxes are named similarly in different as the functionality of a component do not change across the modeling levels. Since the logical components are defined as logical solutions for their allocated functions, they specify generic types of components. The logical component and behaviors are presented in blue boxes. For example, in Table 2-3, the logical component solution for the system function “Generate Mechanical Power” is “Engine”. At physical architecture layer, logical components are transferred from logical to physical architectures and serve as physical component behaviors. Therefore, they are named similar to logical components and presented in blue boxes. These physical behaviors are allocated to physical nodes, yellow boxes, which present the specific types of the components. For logical component “Engine”, the physical node is “Dual-spool Turboprop”. If the specific technical component has not been identified yet, the same name as the logical components is used as a placeholder for physical nodes.

Table 3- 2: Capella nomenclature of elements in different modeling layers for the propulsion system

Modeling layers					
System Analysis	Logical Architecture		Physical Architecture		
System Function	Logical Function	Logical Component	Physical Function	Physical Component Behaviour	Physical Component Node
Propulsion System					
Generate Mechanical Power	Generate Mechanical Power	Engine/ Gas Turbine/ Internal Combustion Engine (generic type or sub-type)	Generate Mechanical Power	Engine/ Gas Turbine	Dual-spool turboprop
–	Transfer Mechanical Power	Accessory Gearbox	Transfer Mechanical Power	Accessory Gearbox	Accessory Gearbox
–	Reduce Shaft Speed	Propeller Speed Reduction Gearbox	Reduce Shaft Speed	Propeller Speed Reduction Gearbox	Propeller Speed Reduction Gearbox
–	Generate Support-Thrust	Supporting Propeller	Generate Thrust	Supporting Propeller	Electrically driven variable pitch
Generate Thrust	Generate Thrust	Propeller	Generate Thrust	Propeller	5-blade MTV-27

According to Figures 3-3 to 3-5, all of the functions and logical components allocated to the propulsion systems are summarized in Table 3-2. The architectural element naming is chosen in a way to be generic and usable among most of the propulsion systems of HE aircraft, except for the physical components. Depending on the type of aircraft and its mission, physical component selections will vary. For instance, according to the ATA (Air Transport Association) 100 chapter 61, specific requirements such as design, performance, and safety requirements need to be considered when choosing the propeller types [97]. In this context, the physical components used for the HE test case are selected based on the reference aircraft (hybridized Do-228) used in Hoffman et al. [18].

As stated above, this thesis aims to enhance the basic Capella specification model to enable the bridge to MDAO and MBSA analyses. As discussed in Section 2.2.2.1, the PVMT is selected to

be used in this study to enrich the Capella models. As part of the AGILE4.0 project, the model specifications in this work are tested and linked to a system-level MDAO workflow by enriching the architectures with particular input parameters for physical components. This is discussed in more detail in Section 3.2. The application of PVMT for building a customized safety assessment property package to support the FHA process is explained in Section 3.3.

3.2 Connecting MBSE to MDAO

The proposed framework seeks to establish a connection between MBSE and MDAO, which would result in a more efficient approach to aircraft systems architecting. Figure 3-7 illustrates the MDAO process within the context of systems architecting. The system architecture design space is depicted as a graph-based descriptor and is evaluated for feasibility based on predefined criteria. This filtering process leads to a reduced set of architectures that are then modeled within an MBSE environment.

MBSE is introduced not only as a means of linking MDAO and systems architecting but also as an integral component of the systems architecting process. The MDAO environment comprises the MDAO specification, which includes details about the tools, interconnections, and choices of MDAO architecture, as well as the designation of different design variables, objectives, and constraints. This information is structured within the MDAO data schema and enables MDAO execution through a Process Integration and Design Optimization (PIDO) platform, as discussed in Chapter 2, Section 2.3. In addition, the model-based specification is expected to facilitate the development process by allowing for the reuse of model artifacts beyond the conceptual design phase. It is also noted that ensuring an architecture specification early in the design process helps establish a traceable baseline for later reference.

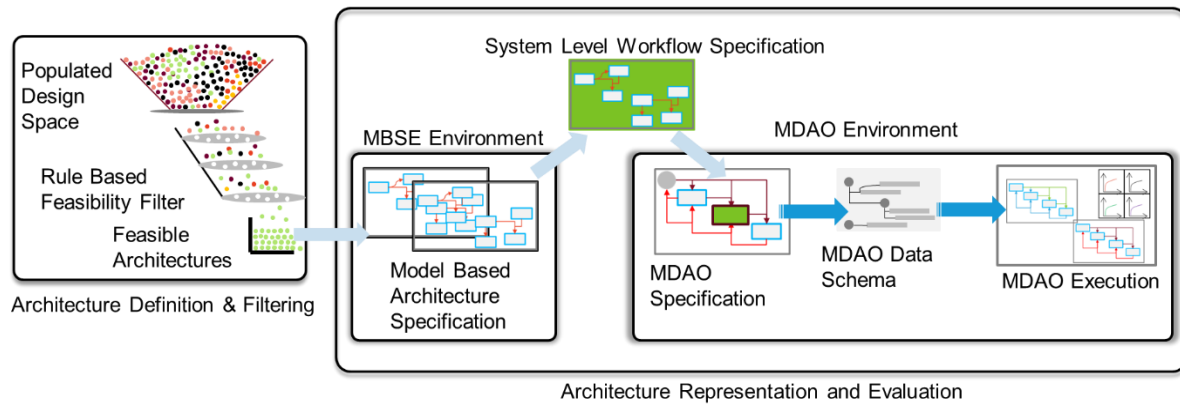


Figure 3- 7: Proposed framework for integrating MBSE and MDAO for systems architecting, from [96]

The remaining few system architectures, after the feasibility filtering process, can be evaluated in an integrated MDAO workflow that operates at both the aircraft and system levels. The information required for the MDAO specification is obtained from the architecture specification model. This architecture specification helps to configure the system-level workflows by considering information from top-level requirements such as range, payload, and mission profile. These subsystem workflows are then formally specified and integrated into an MDAO specification, while the aircraft-level workflow is assumed to have already been created using tools available in the repository.

This specification is then incorporated into an existing MDAO specification database such as CMDOWS. It allows MDAO workflows to be executed within existing MDAO implementation frameworks such as RCE [98] and OPTIMUS [99]. For each architecture, one point iteration is conducted, followed by a comparison of the MDA results. This stage allows architectures that exhibit unacceptable performance characteristics to be discarded. A number of aircraft metrics are considered for evaluation, including maximum Takeoff weight, fuel burn, and operating costs. A comparison of the relative performance of each architecture is conducted based on the optimized results. Using the MDAO results published, a multi-attribute decision-making approach is employed to select the final architecture of the system.

The proposed framework leverages a model-based architecture specification to provide more detailed information on the MDAO specification.

Within the scope of this work, elements of the model are enhanced by incorporating data that quantifies their characteristic properties, which are beneficial for system-level sizing tools.

Examples of such properties include efficiency, depth of discharge, and specific power assignment for a battery in the HE propulsion system architecture. An algorithm is presented in the next section that potentially can automate the process of extracting this information from the underlying data model. The extracted properties are then communicated to the associated tools, and the results are recorded as additional model properties in the Capella model elements.

3.2.1 Model Data Extraction and Connection to AGILE 4.0 MDAO workflow

The study outlines an initial approach to integrating the enhanced model with AGILE 4.0 project workflows [22], as described in Chapter 1. This approach involves using a static workflow, where inputs from the Capella model properties are extracted and provided to the workflow specification data model. It focuses on integrating an architecture specification created in Capella with a system-level MDAO workflow. The Capella model is first enriched with specific input parameters corresponding to tools in the system-level workflow. The enriched model is then translated into a graph-based MDAO specification, which is stored in the Common MDAO Workflow Schema (CMDOWS), along with information on tool interconnections and execution order. A PIDO framework like RCE processes the CMDOWS file to execute the workflow and obtain results. Here, there are three main aspects of this integration using the example of the Flight control system and tools developed in the AGILE4.0 project:

1. Adding system-level tool input parameters to the Capella model
2. Extracting parameters from the Capella data model
3. Integrating the extracted parameters into an MDAO workflow.

Figure 3-8 depicts a set of tools implemented to facilitate the integration of MBSE and MDAO.

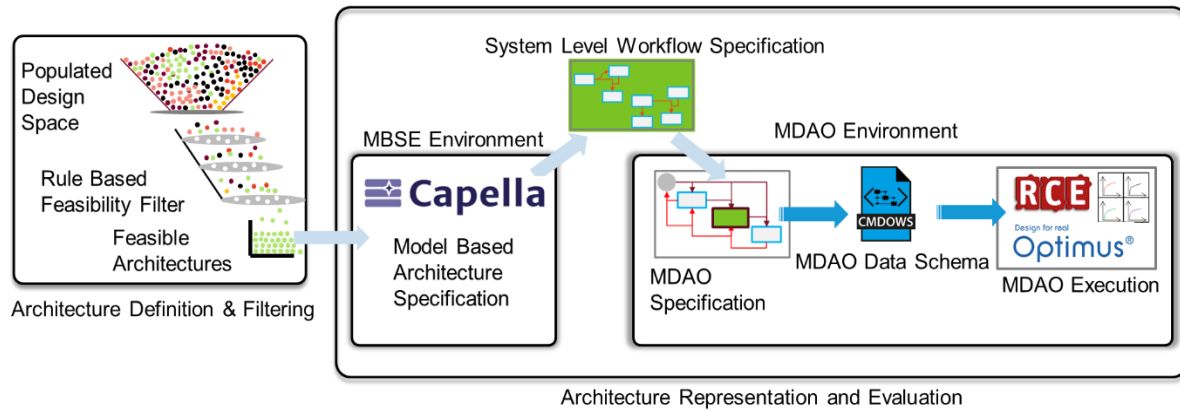


Figure 3- 8: Tool selection for MBSE-MDAO integration framework, from [94]

Parameter Extraction from the Capella Data Model

Capella provides support for various types of information, such as component descriptions, architecture summaries, and model validation information in its architecture specification. However, Capella does not natively support the addition of custom or model-specific properties and the interrogation of model elements to access these properties and needs a plug-in tool for additional capabilities.

Capella is constructed using the Eclipse Modelling Framework (EMF), which can be customized by altering the primary features and creating additional components [100]. The framework includes a structured data model that stores all information related to the Capella model. There are two ways to interrogate the Capella model:

1. Through visual means by utilizing the workbench and diagram editors within the Capella environment
2. Accessing the underlying data model and schema.

To visually interrogate the Capella model, the system architect must manually interact with the diagram editor and access the PVMT viewpoint tab, as discussed in Section 2.2.2.1. Afterward, the system architect will need to manually export and transfer the information to an Excel sheet or directly to the input file of an MDAO system-level tool or workflow.

All data model information is written to two files, which are now called “.capella” and “.aird” in Capella version 5.0.0, previously known as “.melodymodeller”. The “.aird” file contains Capella

project details of active diagrams, such as the formatting and location of elements in a diagram and a log of model changes. On the other hand, the “.capella” file carries information about each model element, including assignments, characteristics, input and output ports, as well as any assigned property values.

Accessing and modifying the underlying data model presents a feasible method to integrate an MBSE specification in Capella with external tools such as Visual Studio Code. The data model is structured in an XML schema that contains comprehensive details about the system architecture specification, such as function, logical and physical component names, functional and component exchanges, and physical links. Additionally, the schema stores data on assigned properties and their respective values, along with the position of graphical elements.

The schema assigns a unique alphanumeric identifier to each model artifact and links between elements reference these identifiers. A simplified schema is created to make navigation easier by only containing the necessary tags for property extraction, with the mapping from the actual schema tags to the simplified version. Figure 3-9 outlines the algorithm for retrieving model properties from the “.Capella” file. This method relies on two key assumptions: firstly, a predetermined list of components is established in Capella, complete with unique identifiers and individual names. Secondly, a list of properties is defined with the PVMT tool, and variable names are specified in the “key” field within the PVMT viewpoint editor [94].

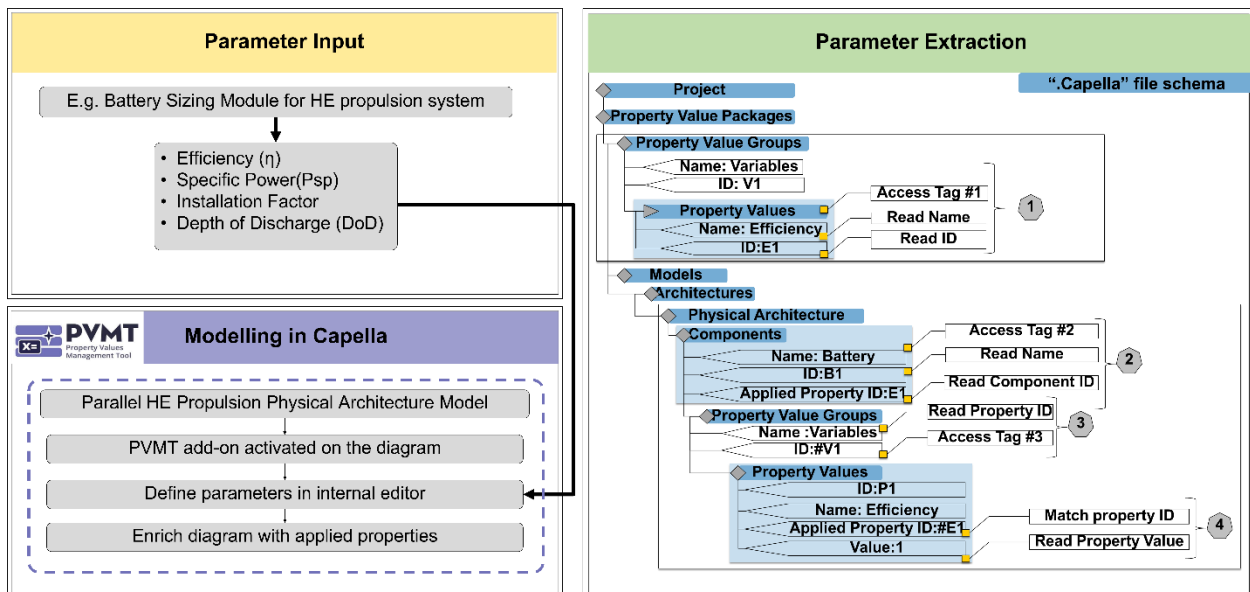


Figure 3- 9: Algorithm of parameter input and extraction in Capella

To begin, inputs are collected and integrated into the Capella model using the PVMT tool. To achieve this, the PVMT viewpoint should be activated within Capella, and the data type for each property must be defined and named. Once properties are established, values are assigned to each field, and the model should be saved. Parameters can be extracted by reading the top-level property value group tag and identifying the name and ID of the child property value tag. Next, the components tag can be accessed, and the name, ID, and applied property ID of each physical component can be parsed. The child property value groups tag can then be accessed, and the ID value can be read. Finally, the property values tag can be accessed, which is itself a child of the property value groups tag assigned to the physical component. At this point, a match can be made between the applied property ID and the property value ID defined in step 1 of Figure 3-9, and the value of the property can be stored [94]. This process can be simplified by creating a simplified schema that only includes the relevant tags, as described earlier. The process of adding and reading component parameter and data from Capella XML source code and mapping from the actual schema tags to a simplified version can be found in Appendix.

Transfer Parameters into AGILE 4.0 Workflows

In the context of the AGILE4.0 project, the MDAO workflow specification is generated through a collaborative effort involving the KADMOS tool and the KE-chain process platform [101], [102]. In KE-chain, each tool's owner defines a "design competence" and initializes input and output parameters using separate CPACS files as an interface standard. After that, a consolidated CPACS baseline file is generated by connecting all the input and output files of the individual tools. This serves as a means of communication between the tools during the execution of the workflow. Following that, the workflow architecture is modified, and the tool execution order is specified. In addition, design parameters are set up, and constraints, objectives, and state variables are defined. This is followed by the generation of a CMDOWS file, which can be used to execute the workflow in RCE, together with the input baseline CPACS file. There are three ways to transfer parameters into the MDAO workflow:

1. directly modifying the CPACS and CMDOWS files
2. packaging extracted parameters and integrating them with the AGILE environment to generate a CMDOWS file
3. integrating the extracted parameters as an example tool.

The third method is the most straightforward but has limitations, such as the need for an existing baseline workflow and only considering system-level tool inputs, which are unchangeable. The process and resulting workflow with the integration of parameters derived from Capella are shown in Figure 3-10.

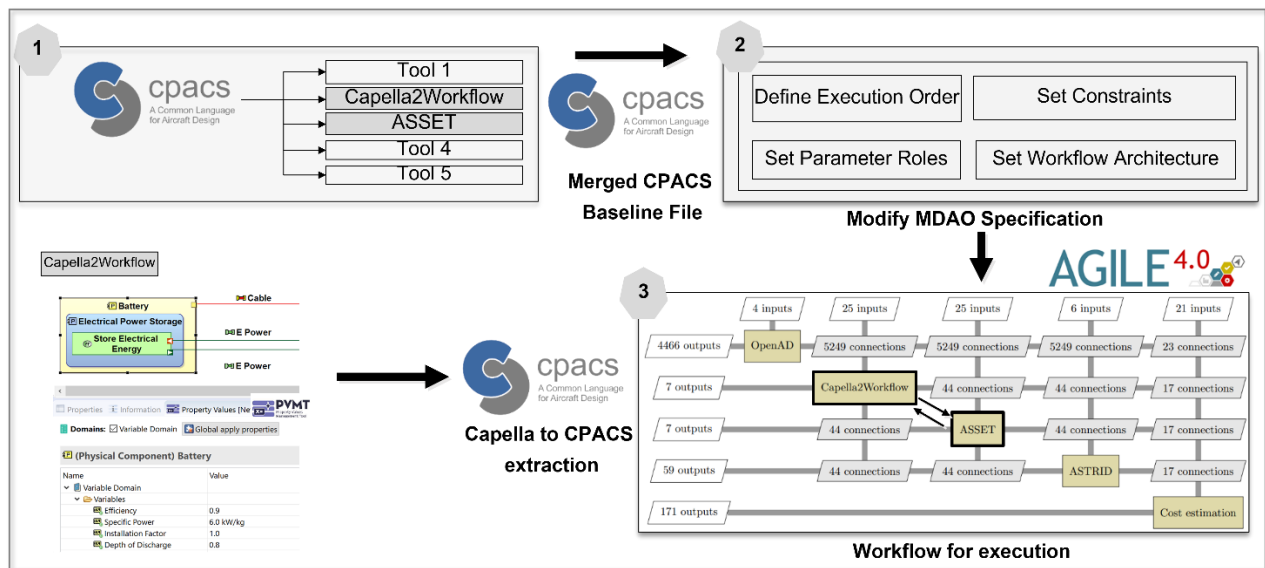


Figure 3- 10: Integrating Capella with AGILE4.0 workflow, from [94]

The ¹ASSET tool [10] takes the system architecture as input to perform component sizing and provides the overall component weight and power demand. This tool relies on the parameters specified in Capella, which are extracted using the Capella2Workflow tool. The parameter extraction functionality is integrated into the AGILE workflow by defining the input and output in the AGILE workbench. Capella2Workflow directly provides the parameter values to ASSET, and the output of ASSET is further passed to the ASTRID² tool [103], which performs the sizing of other systems. The parameters extracted from Capella are written to the output CPACS file of Capella2Workflow, which serves as the input file for the ASSET tool. The workflow output is

¹ ASSET: Aircraft System Sizing Estimation Tool in development at Concordia University

² ASTRID: Aircraft system sizing and performance estimation tool developed by the Politecnico di Torino

written to the baseline CPACS file, and the relevant parameters are extracted and written back to the Capella data model schema, which can be inspected within the Capella workbench.

This section showed how properties could be added to Capella model elements and extracted by parsing the data model schema. The approach to integrating Capella with system-level MDAO tools in the AGILE4.0 workflow is presented, but it requires the system-level workflow to be predefined, and there is a limitation to reconfiguring the workflow based on the referenced tool. Different modeling elements are also used to infer tool inputs, but an algorithm is required to make this process efficient. The study identifies areas of improvement to mature the MBSE-MDAO link functionality [94].

3.3 MBSE to MBSA Integration

Referring to Figure 3-1, this section focuses on integrating the document-based FHA process into the MBSE environment in the early design stage of aircraft at the aircraft level and system level. It attempts to facilitate the FHA output transition to other steps in the safety assessment process, especially for MBSA.

Within the safety assessment process, the Functional Hazard Assessment (FHA) is an early activity that requires functions as input and determines safety objectives, which are crucial inputs for system architecture. Functional analysis is typically a vital component of any MBSE approach, making it a suitable starting point for exploring the relationship between MBSE and a potential model-based FHA [95]. Current industry practices rely on an FHA process that is iterative throughout the design development [23], and as the system modeling and development evolves, the system integration processes should be adaptable to the new methods. Enhancing the model-based system specification to ease the FHA would be a first step towards MBSA [95].

It is important to note that the modeling artifacts used in the MBSE framework are mapped to specific disciplinary tools based on the inputs and outputs that each artifact is assigned. This helps to ensure that the modeling process is streamlined and that the resulting system architecture is well-designed and effective.

The system architectures that are created through the proposed methodology undergo additional levels of evaluation to ensure that they meet safety requirements. Enriching the model with the outputs of safety assessment could then help support individual MBSA analyses such as Preliminary Aircraft Safety Assessment (PASA) and Preliminary System Safety Assessment (PSSA). This includes conducting FHA to identify potential hazards and risks associated with the system. From the results of the FHA, Fault Tree Analysis (FTA) is extracted to further analyze and mitigate potential failures or hazards.

3.3.1 FHA integration within the MBSE framework

The integration of the FHA into MBSE is analogous to integrating behavior and requirements into MBSE. The primary objective of the model-based FHA is to identify safety requirements emerging from the functional and logical architecture of a system. To achieve this, the FHA process is integrated into an MBSE framework, which includes key steps like functional analysis and the synthesis of functional architecture with system architecture modeling. By conducting modeling and FHA simultaneously, safety requirements can be elicited and incorporated into the system model in real-time to enhance them with safety information. This enriched system model then serves as the basis for downstream analysis using MBSA tools to generate artifacts like fault trees and reliability block diagrams. This process includes system architecture specification and the failure identification, classification, and tracking of failure effects. The model-based architecture specification is continuously updated with information from the FHA and the inspection of the model by the system architect, ensuring that the resulting system architecture is well-designed and meets all necessary safety requirements.

System Architecture Specification & Modeling

The proposed MBSE multi-level modeling approach is followed to model system architecture as directed in Section 3.1. Each modeling layer in Capella provides various diagrams with a different objective. Starting modeling at the SA level, the System Functional Break Down [SFBD] diagram generally shows the functional decomposition and relationships. A similar diagram at the LA modeling level shows the Logical Functional Breakdown [LFBD]. There are more common-used

diagrams at the SA and the LA: System/Logical Dataflow Blank [SDFB] and [LDFB]. Typically, the application of this diagram is to illustrate the functional chain and define specific use case scenarios of the system.

After defining functions and the functional exchanges in the SA layer, they are allocated to systems using the System Architecture Breakdown [SAB] diagram. This diagram gives a graphical view of every element identified in the previous diagram, including system functions, allocations, connections, and data flow. A similar diagram is available at the LA layer, i.e., the Logical Architecture Breakdown [LAB] diagram, presenting the logical component exchanges and functional allocations to these components other than [SAB] capability [95].

Moreover, the system engineer can define scenarios in Capella. Functional Scenario [FS] diagrams aim to combine the functional chain with time sequence into one concrete diagram. Aligned with the scenario and functional chain diagrams, the Modes and States [MSM] diagram demonstrates the system's expected behavior in situations foreseen at design time and the operating conditions or status of structural elements of the system. The [MSM] diagram reflects various concepts, such as the phases of a mission or flight (e.g., taxi, take-off, climb) and the specific functioning requirements of the system under certain conditions [42], [104], [105].

3.3.2 FHA Principles & Application

The novel modeling methodology and MBSE environment presented in this study provide graphical and visual model diagrams that can replace predominantly document-based safety assessment processes, specifically the FHA. The primary goal of the FHA is to identify each failure condition and determine its classifications. The FHA process can be divided into two parts, namely, the aircraft-level FHA (AFHA) and the system-level FHA (SFHA). The AFHA defines safety objectives at the aircraft level, including single or multiple system function failures. The output of the AFHA serves as a starting point for the preliminary Aircraft Safety Assessment (PASA). The SFHA is an essential part of the overall aircraft-level safety assessment process, systematically identifying hazards associated with significant system functions and describing

them in terms of operational and functional effects. The output of the SFHA is used to conduct the PSSA [24].

The AFHA takes as its input the aircraft-level functions, which are defined in the aircraft functional architecture and are aligned with the aircraft design requirements and objectives. These functions are typically broken down into sub-functions to create a hierarchical structure. To facilitate this process, Capella is best suited because it has diagrams at the SA layer that are related to functions, and it follows ARCADIA principles. By practicing at this layer's diagrams, the top-level aircraft functions can be decomposed into the lowest level of aircraft sub-functions. The decomposition of lower-level aircraft functions does not necessarily involve assigning functions to systems. Instead, it helps in identifying more detailed failure conditions that can be used as the initial step in identifying failure conditions. It should be noted that in Capella terminology, the aircraft-level functions are referred to as "system functions" at the SA modeling level.

The AFHA in Capella is primarily applied at the SA level and partially at the LA level, focusing on aircraft-level functions. On the other hand, the SFHA is solely implemented at the LA level, covering aircraft systems and their corresponding system-level functions. To effectively integrate both processes, a modeling level such as LA level 0 (L0) can be utilized to break down lower aircraft-level functions into system-level functions using the [LFBD] diagram. Meanwhile, the SFHA can utilize modeling levels L1 (for unconventional aircraft systems) and L2 to allocate system-level functions to the corresponding aircraft systems using the [LAB] diagram.

Figure 3-11 presents the proposed formalization of the entire process supporting the FHA in Capella, which includes several activities such as defining the aircraft-level functions, performing the functional breakdown, foreseeing the functional failures and the potential failure effects, determining severity classification, and assigning flight phase [95].

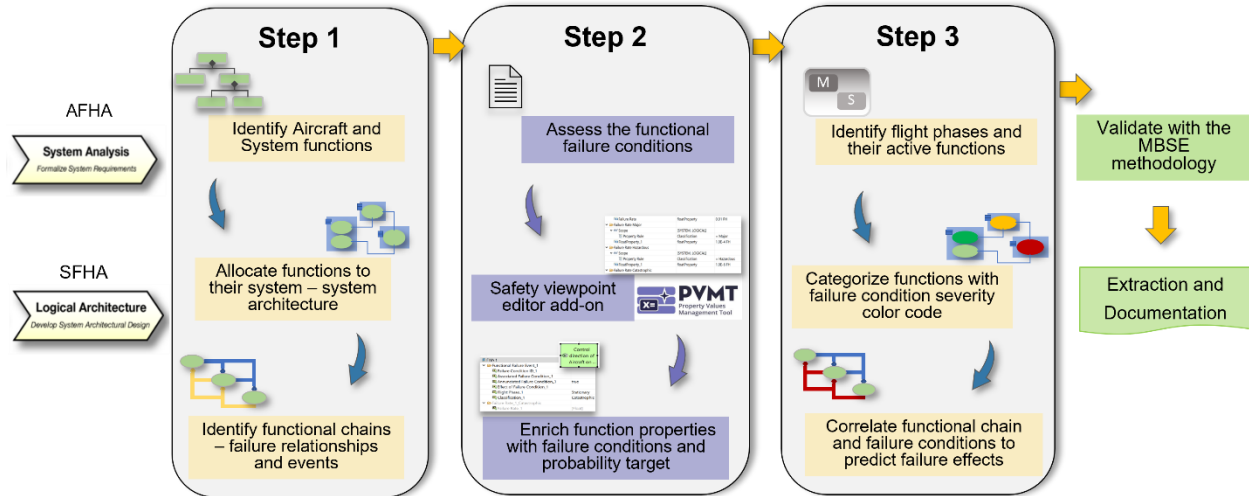







Figure 3- 11: Overview of the formalized process of FHA integration and application within Capella

The procedure outlined in Figure 3-11 is designed to efficiently perform the AFHA and SFHA for each aircraft system using Capella. The steps are as follows:

- Step 1: Define functions using functional breakdown diagrams for aircraft-level and system-level functions at SA and LA modelling levels, respectively. Allocate functions to the system(s) at both levels and create architecture diagrams. Identify primary failure relationships and functional connections to predict potentially affected functions and failure conditions.
- Step 2: Enrich function properties with failure conditions, effects, target average probability of occurrence, and impacted failure conditions of other functions.
- Step 3: Analyze different diagrams to gain better insight into failure conditions and effects. The modes and states diagram identifies flight phases with active functions, while functional chains and scenario diagrams help identify cause and effect of failure events, such as potential combination and propagation effects. Functions are color-coded to visually indicate their most severe failure condition classification in the system architecture.
- Validation and extraction: Validate architectures and modeled systems using the MBSE methodology presented in section 3.1. The system architectures created here, follow the same process as presented in Figure 3-2. Export in image and CSV format and document in Microsoft Excel and Word.

Overall, this formalized process enables the consolidation of predominantly document-based safety assessment processes into a visual and graphical model-based FHA. These activities are supported by the use of PVMT, the modes and states diagram, the SA and LA level diagrams. The implementation of this procedure is summarized in Table 3-3, each diagram serving a purpose in the context of this research.

Table 3- 3: Overview of selected Capella diagram in support of implementing the FHA process

Capella diagram	AFHA	SFHA	Purpose
	System Functional Break Down [SFBD]	Logical Functional Break Down [LFBD]	Create hierarchical Aircraft-level and System-level functional decomposition
	System Data Flow Blank [SDFB]	Logical Data Flow Blank [LDFB]	Define the functional chain for cause and effect failure relationship
	System Architecture Blank [SAB]	Logical Architecture Blank [LAB]	Define function allocations to system(s)
	Functional Scenario [FS]		Supplementary to functional chains for different failure effects realization
	Mode State Machine [MSM]		Define flight phases and their active/critical functions in sequence

The supporting Capella diagrams and detailed implementation of the proposed model-based FHA and its integration with the MBSE methodology for the wheel braking system of landing gears are described and demonstrated in Section 4.2.

Enrichment of the Capella Model Elements to Support Safety Assessment

To facilitate the safety assessment process, starting with the FHA, modifying elements of the system models is proposed to convey information about the failure conditions, their effects, and their severity. Here PVMT is used as a means to enrich the models with safety specifications information relevant to the FHA. The analysis viewpoint modules can assist the system architecture evaluation process in terms of feasibility and safety.

An example of a customized property package created as part of the presented research work for FHA activities is depicted in Figure 3-12. By using PVMT and Viewpoint Editor, as explained in

Section 2.2.2.1, engineers can create a property package in Capella that is tailored to the specific needs of their system architecture and engineering processes and can improve the consistency and quality of the system models and enable more effective analysis and evaluations in several domains. In the Appendix, a step-by-step approach for using PVMT in Capella and creating a safety viewpoint is demonstrated.

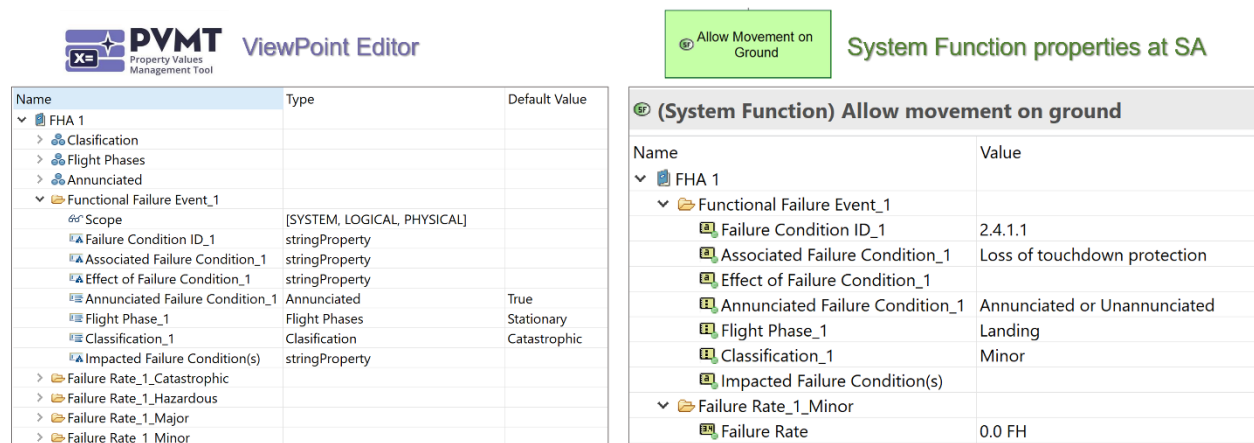


Figure 3- 12: Safety assessment property package defined in Viewpoint Editor by PVMT and applied to a system function at the SA layer in Capella

According to Figure 3-12, a system function, *Allow Movement on Ground*, enriched with data for an FHA viewpoint, including an example of a failure condition ID, failure condition description, the flight phase(s) in which the failure condition can occur, and the classification of failure severity followed by the impacted failure conditions, and the average target probability of occurrence.

Typically, identifying failure conditions involves breaking down the high-level aircraft function. As described earlier, PVMT enriches the properties of model elements by providing a safety viewpoint tailored to the model. While multiple failure conditions may be linked to a function, Figure 3-12 illustrates an example of how failure condition specifications can be assigned to a function. By employing the customized safety assessment viewpoint, one can identify the failure conditions for a given function, assess their impact on the aircraft, and classify them based on severity. This information can be used later to model the propagation and combination of failure effects and to perform other safety analyses. The severity classification for each failure condition is determined by examining the statements of effects in the properties of functions against the relevant regulatory guidance. The classifications, namely Catastrophic, Hazardous, Major, Minor, and No Safety Effect, are assigned based on the most severe failure effects of each failure

condition. The PVMT enables functions to be color-coded based on the associated failure conditions and their classifications using the viewpoint editor, and the function color changes in the model architecture accordingly. Furthermore, an automated assignment of a target average probability of occurrence is implemented, corresponding to the severity classification. It's important to note that the severity of a failure condition may change depending on the flight phase in which it occurs. Therefore, the associated flight phase in the viewpoint is assigned through the modes and states diagram.

In summary, to facilitate the identification of failure effects and targets, the results of the proposed model-based FHA can be utilized in other safety assessment analyses like FMEA and FTA. Overall, the proposed method can enhance the effectiveness of the early development process and promote collaboration between system and safety engineers, which is critical to bringing advanced and intricate aircraft and systems to the market more swiftly.

3.4 Summary

This Chapter described the proposed methodology for improving a model-based system architecting process using an MBSE framework. The methodology includes the following aspects:

- ➔ A multi-level modeling approach to structure model and manage model complexity, in-line with the ARCADIA method in the Capella tool
 - ➔ Creating re-usable elements to reduce development time when dealing with architecture variants
 - ➔ Model enrichment and model information extraction using the PVMT add-on
1. Linking MBSE to MDAO workflow to facilitate the use of MBSE in the aircraft design process from the conceptual design stage, enabling more comprehensive system analysis, including safety analysis, based on architecture models. This approach paves the way for better integration of MBSE in the aircraft design process.
 2. Integrating MBSE to MBSA by proposing model-based FHA, which enhances the efficacy of the initial development process and fosters better collaboration between the safety engineer and system engineer. This collaboration is essential for expediting the introduction of novel and intricate aircraft and systems to the market.

In the following chapter, the model integration and applications of the proposed methodology are presented, which offers a practical perspective on multi-level modeling specifications and the integration of model-based FHA to the MBSE framework.

4 Model Integration and Application

The previous chapter proposed a methodology for system architecture representation within the MBSE framework and explored the connection to MDAO workflow. Moreover, it introduced a model-based FHA as an initial step toward MBSA.

This chapter presents two test cases that apply the developed framework. The first test case illustrates the multi-level specification framework for aircraft and system-level models, including the following systems: the hybrid-electric propulsion systems, the yaw control system (as part of the primary flight control system), the hydraulic power system, the fuel system, and their interfaces. In this chapter, system architecture specifications, and configurations are discussed in detail. The second test case focuses on the model-based FHA and is developed for the wheel braking systems.

Each test-case will be preceded with a short description of the system that this modeled to enable a better understanding of the specification models.

4.1 Test case 1: Multi-level system specification for the hybrid-electric propulsion system and associated subsystems

4.1.1 Hybrid-Electric Propulsion System

Before explaining the specification model for the HE propulsion system, a short literature review presents the various hybrid-electric propulsion configurations. This analysis was required to build a generic model and to structure the specification model properly.

The following literatures are used to establish the system specification models. De Vries et al. [106] develop a methodology that utilizes component-oriented constraint diagrams to provide a visual representation of the design space, while Harish et al.[36] use a framework to evaluate different hybrid-electric propulsion system architectures, which considers power distribution, power transmission, and the propulsion system's efficiency. Finally, Hofmann et al. [18] have analyzed the impact of hybridization on a Dornier 228 aircraft and found that a parallel hybrid-

electric architecture can improve takeoff performance and reduce noise footprint but may decrease passenger capacity.

Several types of hybrid-electric configurations exist [107], as the main categories are shown in Figure 4-1. However, only three powertrain configurations for hybrid-electric propulsion are considered as use cases for this work.

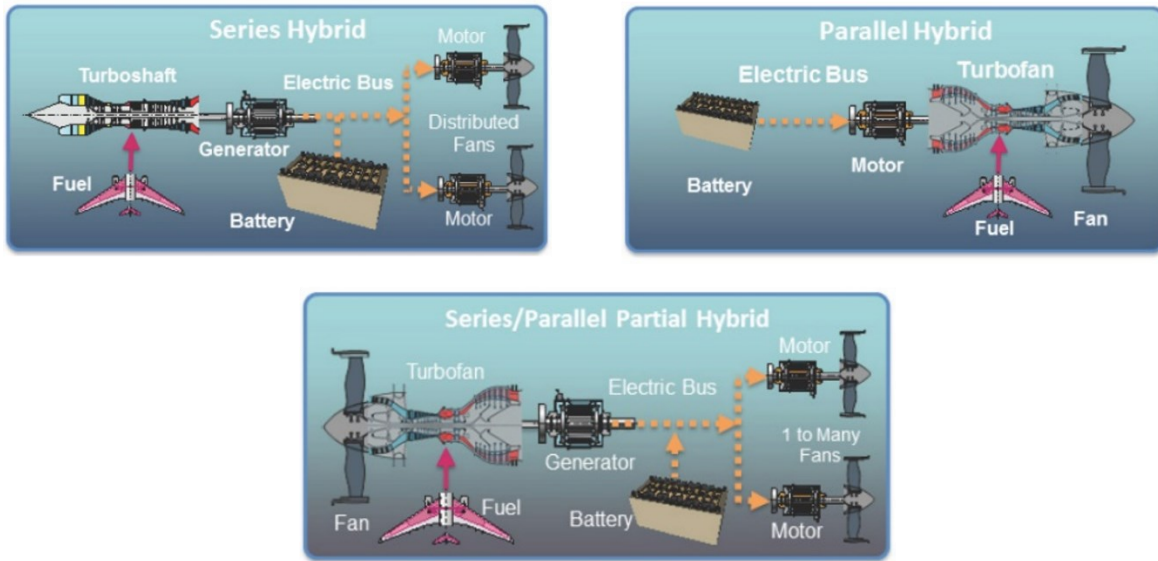


Figure 4- 1:Hybrid- electric propulsion architectures, extracted from [107]

The hybrid propulsion systems utilize gas turbine engines to power the aircraft and recharge the batteries. The batteries can also provide energy for the aircraft's propulsion during one or more stages of the flight. In Figure 4-1, a parallel hybrid system consists of a battery-powered motor and a turbine engine, both mounted on a shaft that drives a fan, allowing for either or both to provide propulsion at any given time. On the other hand, in a series-hybrid system, only the electric motors are directly connected to the fans, while the gas turbine drives an electrical generator, which then powers the motors and/or charges the batteries. Series-hybrid systems can be applied in distributed propulsion concepts using multiple small motors and fans. The series/parallel partial hybrid system has one or more fans directly driven by the gas turbine and others exclusively driven by electric motors. These motors can be powered by a battery or a turbine-driven generator [107].

Series Hybrid-Electric Propulsion Configuration

In the series architecture, fans are solely driven by electric motors supplied with electricity by a generator. An internal combustion engine (turboshaft) feeds mechanical power to the generator, the output of which drives the motors and/or charges the batteries usually added to the powertrain to act as a barrier in peak power demand. The savings (charging batteries) are accumulated when high energy density fuel is burned during the cruise flight [107], [108].

Although the main energy source comes from hydrocarbon fuels, which increases payloads and ranges, the dual energy conversion is less efficient than the single one due to the conversion from fuel to mechanical energy and then from mechanical energy to electric energy. For instance, 15% of the energy will be lost between the gas turbine and the propellers, considering efficiencies of 95% for each electrical machine and 97% for each power electronic [18]. Therefore, combining the gas turbines into a single, more efficient unit placed inside the fuselage is more logical and applicable, which consequently increases efficiency and reduces noise. However, it will involve heavy electric machinery bringing weight penalty to the series system configuration. As shown in Figure 4-2, series hybrid systems could use multiple relatively small motors and fans, meaning that distributed propulsion concepts can be applied through the series configuration. To this end, setting the number of propellers at four would be decent according to previous design research [18], [108].

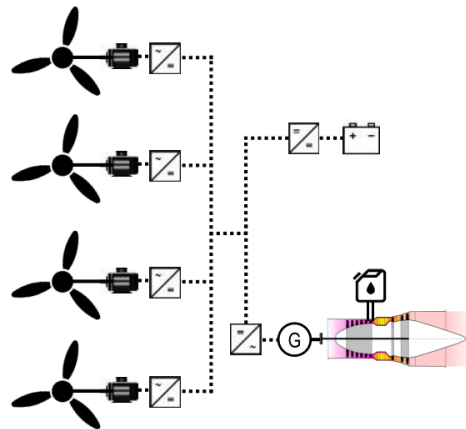


Figure 4- 2: Series hybrid powertrain, from [18]

Parallel Hybrid-Electric Propulsion Configuration

In this configuration, a battery-powered motor and a turbine engine are both set on a shaft that drives a propeller so that either or both can run the propeller(s). Contrary to serial-hybrid powertrains, parallel-hybrid powertrains include single-energy conversion. They are categorized in two various arrangements. One possible architecture is “Torque Split”- Parallel, in which the mechanical power from both an internal combustion engine and an electric motor can be simultaneously transmitted to a propeller drive shaft via a gearbox. The other one is referred to as “Through the Road”- Parallel (TTR-parallel), a configuration where these two mechanical power providers separately feed two or more different fans or propellers. In this case, some propellers can be run by gas turbines, whereas others are driven by electric motors. In a TTR-parallel layout, there is no link between conventional and electrical powertrain [108]. With the benefit of requiring less numbers of components, this configuration enjoys advantages related to weight saving. Nonetheless, on the downside, it involves mechanical coupling which causes operational and control complexity. Similar to the series configuration, the electrical power can be stored in a battery. The battery can either be charged onboard or on the ground, to supplement the propulsive power in different phases of the flight such as cruise and descent [109]. Hofmann et al [18] recommend to place the propellers at the rear of the fuselage that one may shorten the connection of powerpack and propellers besides the reduction of noise by shielding its blades. Besides, another parallel configuration has been noted which contains two gas turbines instead of one as shown in Figure 4-3. This architecture not only enables achieving the maximum required power in case of failure of one gas turbine, but also reduces the power peak of each turbine in the normal condition. However, this configuration is faced with a weight drawback and is unlikely to be considered for future studies.

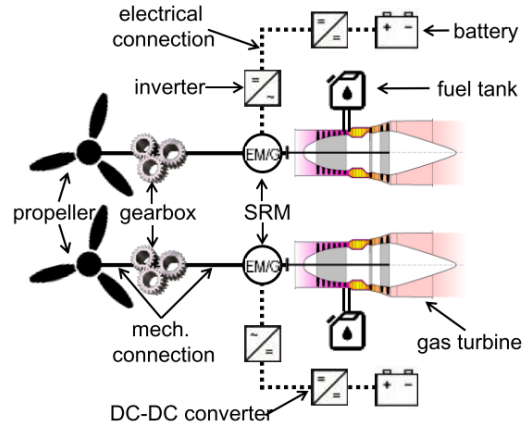


Figure 4- 3: Parallel hybrid powertrain, from [18]

Partial Series/Parallel Hybrid-Electric Propulsion Configuration

A serial/parallel hybrid electric configuration represents the co-existence of both serial and parallel operation. It has one or more main propellers, which are mechanically driven by a gas turbine directly as well as smaller fans driven by motors holding electrical power sourced from a gas turbine-driven generator or a battery [107]. One of the advantages that series/parallel hybrid system architectures offer is having lower installed electrical power and, in turn, smaller-sized electrical components since a significant portion of the propelling power is conducted from the gas turbine to the propellers. Restrictions of sizing and operating the engine to satisfy the propeller's high demand power could be lower to enable performance optimization of the engine in various operating segments [18], [109].

Additionally, the combination of parallel and series is the most promising configuration regarding the environmental issues and the performance of expected hybrid systems. Although this combined architecture suffers from electrical power losses due to double energy conversion, they are significantly low compared to the series architecture. Since this configuration, to a great extent, can cover the drawbacks of both series and parallel system architectures, it is used as the reference for the use case HE system architecture within the proposed framework. This configuration is schematically outlined in Figure 4-4.

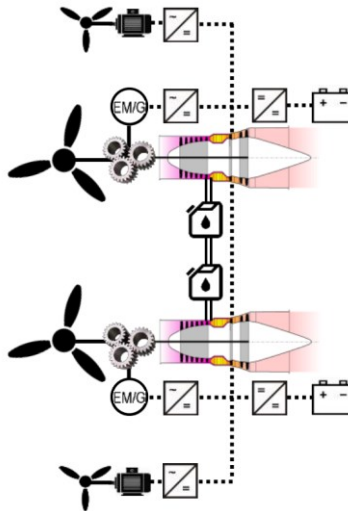


Figure 4- 4: Series/Parallel hybrid powertrain with additional electric boost, from [18]

The following section will discuss the model architectures for series, parallel, and partial series/parallel configurations at the LA modeling layer in Capella. Also, a generic model that covers the configurations of the HE system mentioned above is presented at the SA modeling layer. Other models, such as L1, L2, and P2, are presented for the “Series/parallel with additional electric boost” configuration to avoid repetition.

Capella specification models

Architecture representations following the proposed MBSE approach are performed in Capella. As discussed in Chapter 3, the architecture modeling in this work begins with the SA level.

Initially, the HE system’s generic architecture (S0) is developed at the System Architecture (SA) level. Next, the system configurations of the HE system are modeled generically at the Logical Architecture (L0) level. Then it is followed by creating series/parallel system configuration models at detail levels 1 (L1) and 2 (L2). Finally, a physical model at level 2 (P2) of the HE configuration is developed at the Physical Architecture (PA) level.

It is important to note that the reference model for the presented HE propulsion and electrical system architectures is based on the study by Hofmann et al.[18]for the Dornier 228, a 19-passenger commuter aircraft.

S0: Generic Model at the System Analysis Level

This level of representation utilizes generic components to create a simple architecture of any HE system configuration, focusing on the interactions and exchanges between its components. This model aims to provide a simplified, high-level view of the system's structure and behavior, which can be used to inform early-stage design decisions and facilitate communication between different stakeholders.

By using generic functions, the S0 model allows for a more flexible and adaptable modeling process, as specific details and functions can be added or removed as needed. This approach is particularly useful for complex systems where the details of individual components of the system may not be fully understood or where there is a need to explore different design options quickly.

As depicted in Figure 4-5, top-level aircraft functions related to HE cases, such as “Generate Mechanical Power” and “Generate Thrust,” are allocated to the System of Interest (SoI) “Aircraft.” These top-level functions can be broken down into sub-functions, later in detailed modeling levels. For example, “Store energy” is the parent function for fuel tanks and batteries. Also, at this point, functions are not allocated to any specific aircraft systems at this modeling level. It is to be noted that the aircraft functions depicted in Figure 4-5 are incomplete, and other top-level functions are not covered within the frame of this work.

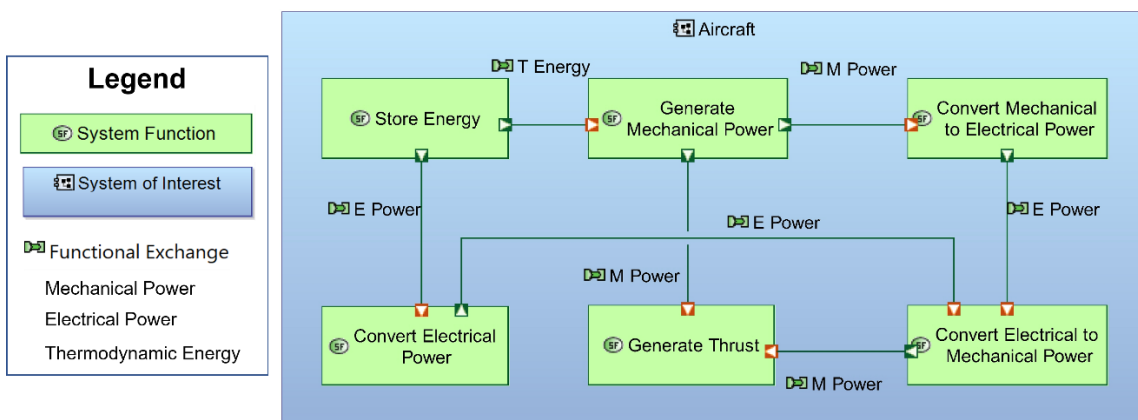


Figure 4- 5: S0-Generic system architecture for hybrid-electric aircraft without interfaces systems at SA in Capella

Functions and their connections in model S0-Generic are arranged in a way to be adaptable for the three types of HE configurations, as discussed in the previous section. Therefore, from this model, all three HE configurations can be derived simply by switching one or two functional exchanges.

This level of representation provides a useful starting point for developing more detailed models of a system's architecture, allowing for a better understanding of its behavior and supporting the development of effective design solutions.

L0 Model at the Logical Architecture Level

The L0 models are created with level 0 detail, as explained in Section 3.1.2, which is similar to the S0-Generic representation. The L0 model allows the HE system architecture representations at a high level. By reusing generic elements, this can be accomplished rapidly and allows for the creation of a broad range of architectures. Nevertheless, more detailed representations are frequently necessary to examine specific interactions within a system architecture or to advance a chosen system architecture. The formation of architecture specifications necessitates a more precise definition of certain elements of the system architecture. This can be accomplished using the L1, and L2 representation levels, depicted in the next sections.

The L0 model aims to allocate the top-level aircraft functions defined in the S0-Generic model to their related systems. At the LA modeling stage, all the systems of the SOI are defined as Logical Components or Logical Systems. Because the SOI is big, only sub-systems are presented in the models.

Moreover, the L0 model illustrates system allocations to other systems, as shown in Figure 4-6. The HE system contains a "Propulsion system" and an "Electric Power Generation and Distribution System". These systems are shown to be connected to other systems, such as the fuel, hydraulic, and flight control systems, through a series of interfaces and connections. This allows for a more detailed analysis of the interactions and exchanges between the different components of the system, which can be used to inform design decisions and identify potential issues.

According to Hofmann et al., partial series/parallel HE configuration is the most promising, while it is more complex. Figure 4-6 generically features this configuration for hybridized Dornier 228 [18]. As shown in the architecture model, there are three ways to provide mechanical power to the "Generate Thrust" function and run propellers.

1. Provide Fuel → Generate Mechanical Power → Transfer Mechanical Power → Generate Thrust
2. Provide Fuel → Generate Mechanical Power → Transfer Mechanical Power → Convert Mechanical to Electrical Power → Distribute Electrical Power → Convert Electrical to Mechanical Power → Transfer Mechanical Power → Generate Thrust
3. Store Electrical Energy → Distribute Electrical Power → Convert Electrical to Mechanical Power → Generate Thrust.

The first and second power chain, rely on “Generate Mechanical Power” and burning fuel, while the third, the greener way, solely depends on electrical energy and it.

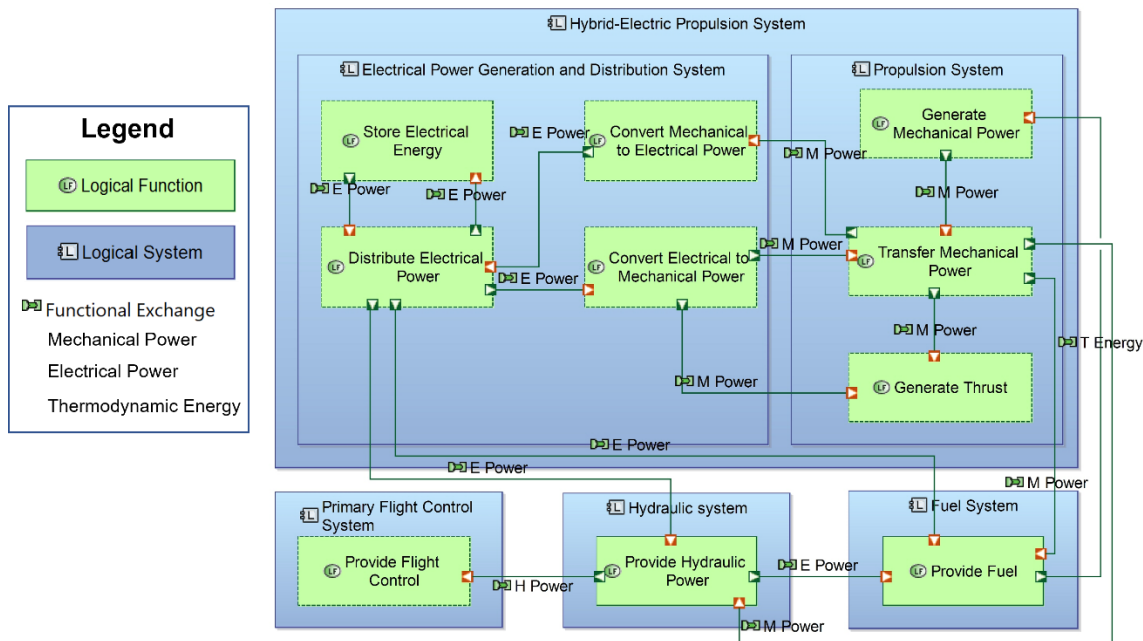


Figure 4- 6: L0-Generic for Parallel HE configuration at LA in Capella

The L0 model for series HE configurations is presented in Appendix.

L1 Model at the Logical Architecture Level

This level of detail model (L1) includes aircraft-level redundancy and technological choices such as the number of engines, number of hydraulic systems, flight control system technology, and level of electrification. The L1 model is necessary for designing unconventional architectures, especially

for HE aircraft systems that are relatively new and unexplored. In the early design stages, by considering technological factors, designers can optimize the aircraft's performance and ensure that it meets the requirements of the intended mission.

Aircraft-level redundancy refers to the use of multiple independent systems to perform critical functions on an aircraft. This redundancy is designed to ensure that the failure of one system does not result in the failure of the entire aircraft. For example, an aircraft with multiple engines can continue to fly even if one engine fails. Similarly, redundant hydraulic systems can ensure that the aircraft's control surfaces remain functional in the event of failure. Figure 4-7 illustrates the model L1 relying on two "Gas Turbine" to provide mechanical power for primary and secondary power generation. For primary power generation, both engines are driving the "Propeller Speed Reduction Unit" which reduces the engine shaft speed to a suitable speed to run the "propeller_1". In secondary power generation, the two gas turbines provide mechanical power for "Accessory Gearbox" to run Engine-Driven Pumps (EDP) of hydraulic and fuel systems. The cross-connections of engines to "Transfer Mechanical Power" function lower the risk of failure of providing mechanical power for aircraft systems.

Moreover, the level of electrification on an aircraft can impact its safety and performance. Electrical systems are often used to power critical components such as flight control systems. More advanced aircraft designs may rely more heavily on electrical systems, such as using electric motors for propulsion instead of traditional jet engines. However, the use of electric systems also introduces new safety concerns, such as the risk of battery fires or electrical failures.

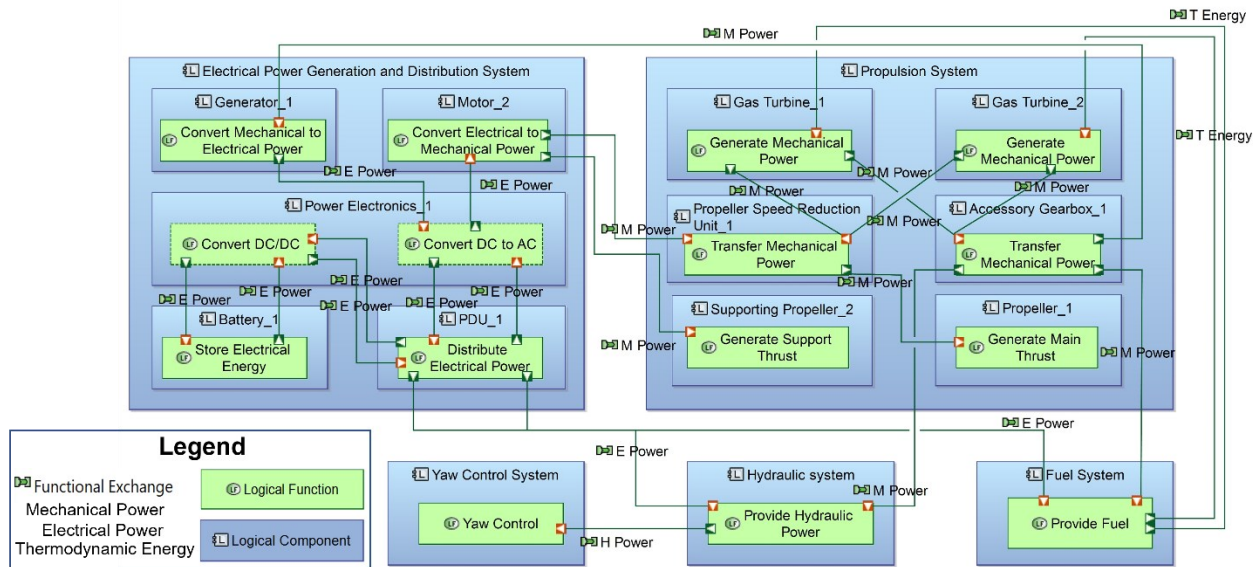


Figure 4- 7: L1 model for partial series/parallel HE configuration at LA in Capella

In both Figure 4-6 and Figure 4-7, the power chain is similar, which is for partial series-parallel configuration. However, the L1 model is more detailed compared to the L0 model. Logical systems are decomposed into logical components. Meanwhile, Top-level functions associated with critical technological decisions are also decomposed into lower-level aircraft functions. These functions are allocated to their related logical system. As types of propellers should be addressed early in the conceptual design stage, the functional decomposition of “Generate Thrust” would be an obvious example of this, which is broken down to “Generate Main Thrust” and “Generate Support Thrust”. Each child function is allocated to distinguished logical solutions, which are “Propeller” and “Supporting Propeller”, respectively.

Referring to Figure 3-1, as discussed in Section 3.1.2, the L1 model is a good starting point for creating system-level models and deriving system-level functions. Section 4.3 illustrates system-level models such as Sys-L1, Sys-L2, and SysP2 for yaw control of the flight control system.

L2 Model at the Logical Architecture Level

L2 model representation provides greater detail on specific components of the architecture where component redundancy is included. A representation of hydraulic and fuel systems and yaw control surfaces is also provided, as they have major interfaces with propulsion and electrical

systems. The interface systems are sufficiently elaborated to show allocations and interfaces to the HE propulsion and electrical systems. By providing a detailed view of the power system architecture alongside the HE propulsion system, it becomes possible to investigate redundancy and to study the functional relationships between the HE system and FCS, and to perform safety analysis in early design stage. The systems specifications and test case assumptions are illustrated in Figure 4-8. The reference aircraft used for yaw control, hydraulic, and fuel systems is Bombardier Challenger 605 business jet [110-112]. The reason for this selection is that the consumer systems have simple architecture and detailed design inputs for system architecture are more accessible and open source.

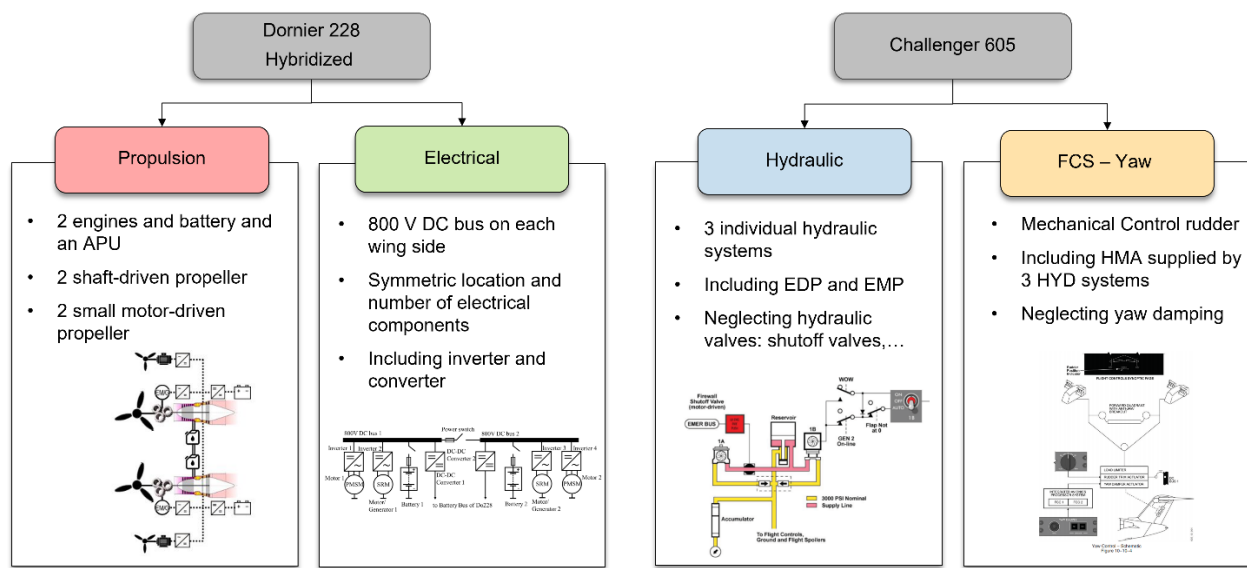


Figure 4- 8: Specification of systems illustrated in L2 and P2 models, adapted from [18], [110], [111]

Figure 4-9 illustrates the application of L2 in the representation of the hybridized Do228's propulsion and electrical architecture. The focus of the L2 modeling representation is on logical component behavior and functional allocations. Also, the connections between the HE system and other systems are well-addressed. This approach also allows designers to focus on the behavior of individual systems and their interactions with other systems without getting bogged down in the details of individual components. The L2 model represents different aircraft systems using dashed line boxes, and the components are categorized according to the system they belong to.

At this level, no information regarding the component technology is given. Basically, system physical specifications belong to the PA modeling level. However, the L2 model features typical

logical solutions for functions. For example, there are three actuators in the yaw control system, but the type of actuators is not specified.

This is important to zoom in on specific interfaces and internal components or to develop certain architectural elements further, with more detail. In this application, the L2 model representation is used to capture HE systems along with interface systems in more detail. The sources of different types of power (e.g., electrical, mechanical, and hydraulic) and how they are distributed to other systems such as primary flight control and fuel systems are outlined in L2 model.

As more representation elements are added, the diagram's readability decreases. While the Capella viewer allows for zooming in and out, this function is not available when the diagram is exported as an image. As a result, practical recommendations were discovered for the efficient export of diagrams, which entail consolidating all the elements together to reduce empty space.

The arrangement of components in the L2 model is symmetrical, which can facilitate navigation and allow for the identification of component redundancy while maintaining an understanding of the overall structure. The symmetrical arrangement of components can help designers to visualize how different parts of the system interact with one another and identify any redundancies or inefficiencies in the design. This approach enables users to zoom in on specific interfaces and internal components while still keeping the big picture.

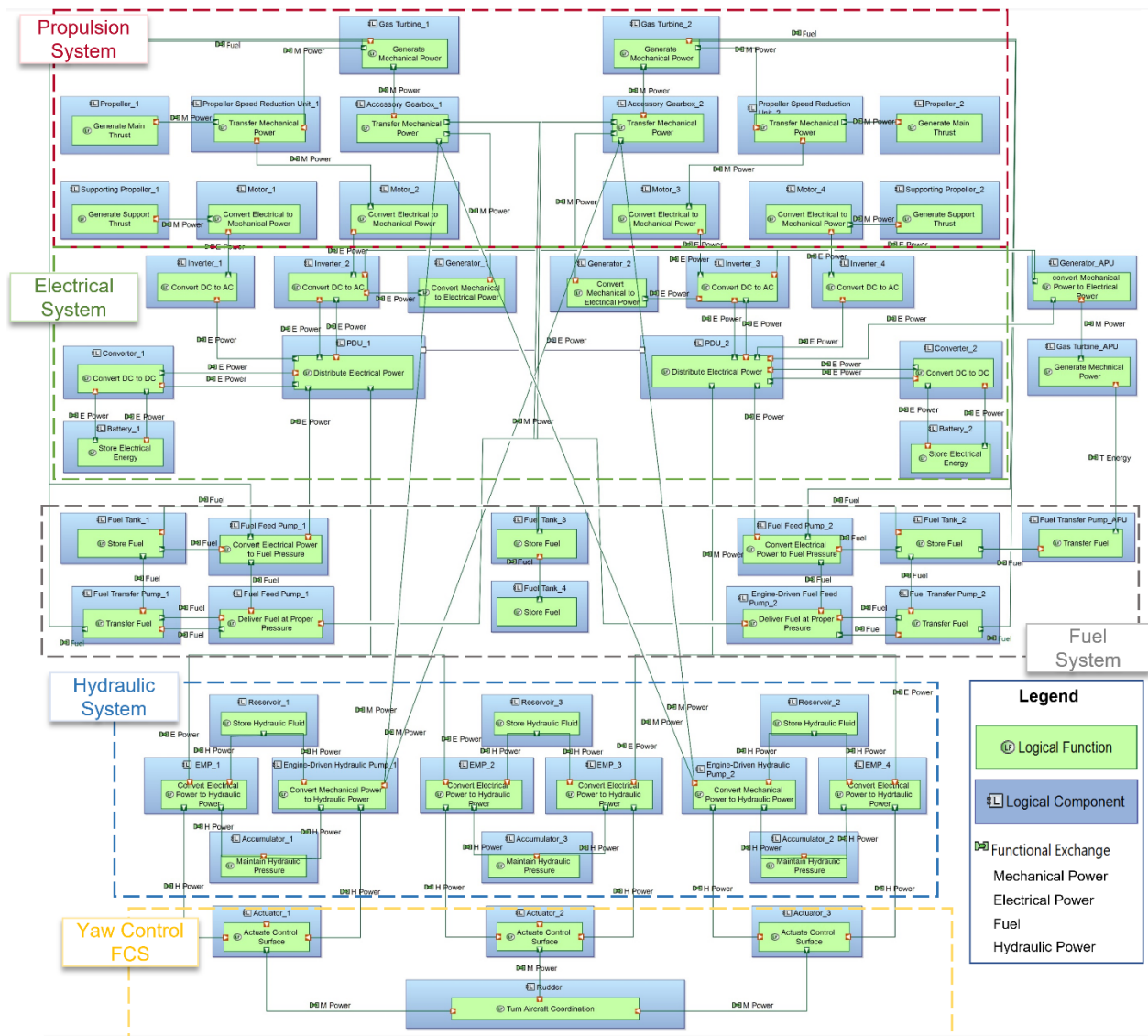


Figure 4- 9: L2 model for partial series/parallel HE configuration in Capella

Identifying the sources and consumers of electrical power is crucial, given the increasing importance of electrical power in both hybrid-electric and all-electric aircraft to ensure safe and efficient operation. The electrical power feeding hydraulic, fuel, and yaw control, has several power generation and distribution elements, which are as follows:

1. Battery: is a rechargeable energy storage device that provides electrical power to the electrical system when the engines are not running or in the event of power shortage.
2. GEN: refers to an electrical generator that derives power from the engine power offtake.

3. APU GEN: refers to an electrical generator that derives power from the Auxiliary Power Unit (APU) engine power.
4. PDU: Power Distribution Unit (PDU) provides primary power bus connectivity, transfer, and distribution functions to the aircraft loads. The default mode of operation for PDU allows for the minimum requirement of supplying power from batteries and generators to the necessary loads. This ensures that the emergency and essential electrical systems remain powered.

As for consumers, electrical power is used for a variety of systems and components in aircraft, including:

1. EMP: or Electric Motor Pump pressurizes the hydraulic system by using the electrical power distributed by PDU.
2. Fuel Feed Pump: deliver a constant flow of fuel to the engines at the required pressure that prevents engine failure due to fuel starvation.
3. Battery: in recharging mode, this component consumes electrical power when the power demand is relatively low.

Also, mechanical power for running propellers and driving mechanical pumps in fuel and hydraulic systems is provided and transferred by the following elements:

1. Gas Turbine: a type of engine that uses a continuous combustion process to convert fuel into mechanical energy.
2. APU gas turbine: a small gas turbine engine used to provide electrical power and/or hydraulic pressure when the main engines are not operating. It is typically located in the tail section of the aircraft.
3. Accessory gearbox: a component that is used to transmit power from the main engine to various aircraft systems, such as the electrical and hydraulic systems.
4. Propeller speed reduction unit: a gearbox that is used to reduce the speed of the engine output shaft to a speed suitable for driving a propeller.
5. Motor: refers to an electric motor that is used to drive various systems, such as hydraulic pumps or fuel pumps.

The consumers of mechanical power in model L2 are mostly Engine-Driven Pumps in hydraulic and fuel systems connected to the engine through the “Accessory Gearbox”. There are two engine-

driven hydraulic pumps powered by both engines, meaning that in case of failure of one engine, the other engine would be able to supply hydraulic system which is a crucial power for FCS. The three hydraulic systems in model L1 power the yaw control surface actuators. Systems 1 and 2 each have one EMP and one EDP, and system 3 has two EMP. This combination of hydraulic pumps prevents failure of the hydraulic system in the event of failure of electrical power or mechanical power sources.

P2 Model at the Physical Architecture Level

The P2 model features the detail level representation the same as the L2 model, as they are both at detail level 2. This model aims to illustrate the physical implementation of the logical components employed in the L2 model. These components are reusable throughout Capella by the REC/RPL capability explained in Section 3.1.

Figure 4-10 captures system interfaces and exchanges along with physical components at the PA level. The system architecture enhanced with physical implementation can be provided to suppliers for development. The selection of HE physical implementations for the P2 model is based on the proposed components by Hofmann et al. [18] and Do-228. Also, the Challenger 605 manual [110-112] is used as a reference for yaw control, hydraulic, and fuel systems component selection. With this model, the exact type of component deployed to the architecture is identified. For example, the technology of the actuator used for the rudder is Hydro-Mechanical Actuator (HMA), and this is according to the selection of Challenger 605 [110]. However, the objective of the P2 model is to focus on the elaboration of the physical architecture of the HE system and interface with other systems. A catalog of the physical components employed in the P2 model is listed in the Appendix.

The model presented in this section emphasizes on aircraft-level functions and shows the functional decomposition in different granularity levels. However, the system-level functions are also covered within this framework which is illustrated for a test case in the following section.

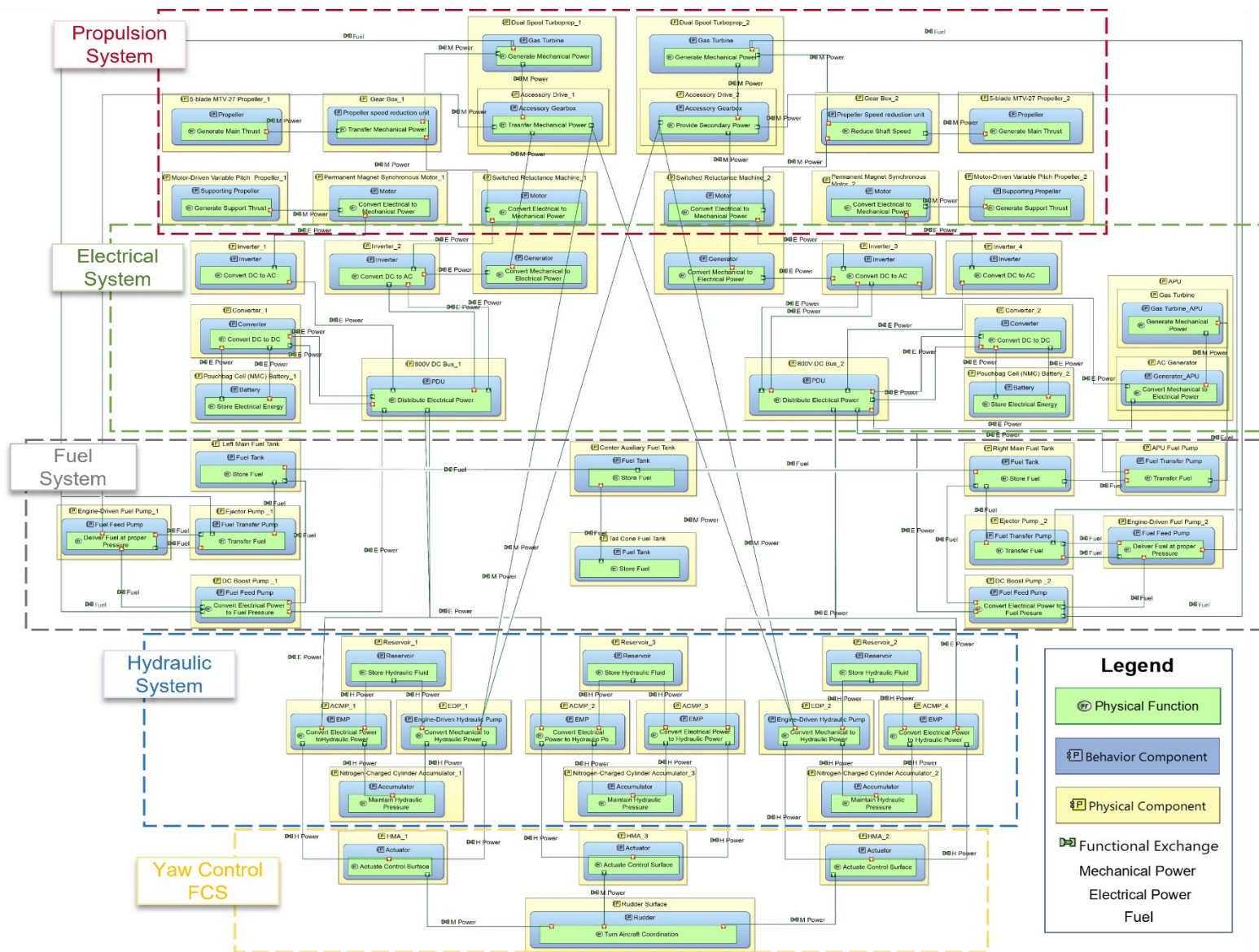


Figure 4- 10: P2 model for partial series/parallel HE configuration in Capella

4.1.2 Primary Flight Control System – Yaw control

In this work, the yaw control system, which is part of the primary flight control system, has been chosen as an example to illustrate the application of the methodology for system-level functions. Although the yaw control system does not have a direct interface with the HE system, it is indirectly dependent on the HE system through the hydraulic system. This highlights the interconnections of different systems within an aircraft and the importance of considering these interdependencies when designing and analyzing aircraft systems. By using the yaw control system as an example, this work demonstrates the importance of taking a system-level approach to aircraft design, considering the interactions between different systems, and identifying potential areas for improvement from a safety perspective.

The Sys-L1 model, which is the first model developed in this study at the system-level, is derived directly from the L1 model at the aircraft-level, by using horizontal adaptation, as illustrated in Figure 3-6. The elaboration principles used in Sys-L1 are similar to those used in L1, but Sys-L1 includes system-level functions with a particular focus on yaw control. Yaw control system in Challenger 605 has mechanical controls through pedals to operate the rudder. Figure 4-11 provides a schematic of the Challenger 605 yaw control system. To simplify the architecture and focus on key features, the yaw damping system has been neglected in the Sys-L1 model. This approach allows for a clearer representation of the system-level functions and their relationships, which can be useful for further development and optimization of the yaw control system.

The yaw control system is designed to provide control over the aircraft's movement around the vertical axis, and it relies on a hydraulic actuated rudder hinged on the rear spar of the vertical stabilizer. The conventional arrangement of the primary flight controls includes rudder pedals for both the pilot and copilot. The cockpit controls transmit movement to the rudder through mechanical means such as cables and/or pushrods, which then use hydraulic power to move the control surfaces. The rudder pedals are connected to three independent actuators through two separate mechanical control paths. Each actuator is powered by a separate hydraulic system, which ensures redundancy and improves the reliability of the system.

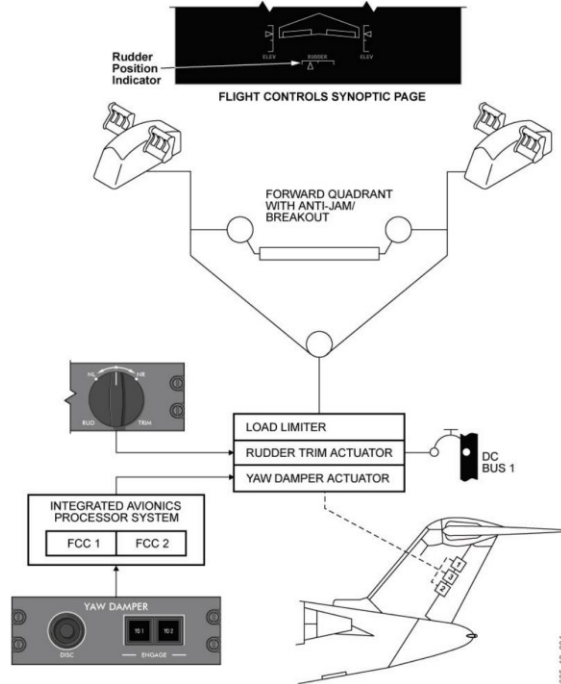


Figure 4- 11: Challenger 605 yaw control schematic, from [110]

Artificial control loading is implemented at the individual rudder pedals to provide pilots with a tactile sense of control. This approach allows pilots to feel the response of the control surfaces and adjust their inputs accordingly, contributing to safe and effective aircraft operation. The rudder pedal control systems are equipped with an anti-jam breakout mechanism that allows for continued operation in the event of a jammed rudder control on one side. This mechanism enables both the pilot and copilot rudder pedals to operate the remaining control system. However, additional force on the rudder pedals is required to bypass the inoperative side, allowing for continued rudder operation. This feature ensures that the aircraft remains controllable in the event of a system failure, contributing to the overall safety and reliability of the aircraft [110].

Figure 4-12 illustrates the Sys-L1 model at the LA modeling level, which places a greater focus on the yaw control system. The HE-related systems are presented with top-level functions and are not decomposed in this model. By using the Sys-L1 model, the detailed connections between the interface system and the yaw control system can be identified without losing sight of the overall

system architecture. This approach allows for a more targeted analysis of the system-level functions and their interrelationships.

Similar to the rules followed in detail level 1, the Sys-L1 model only includes redundancy for those components that are part of the technological choices. For instance, the decision to use three hydraulic systems to provide hydraulic power to the actuator was made early in the design process and is reflected in the Sys-L1 model. However, this model does not present actuator redundancy since it was not considered part of the technological choices. Focusing on critical components and their redundancies, this model provides a more targeted analysis that can help identify potential failure points and improve the overall system reliability.

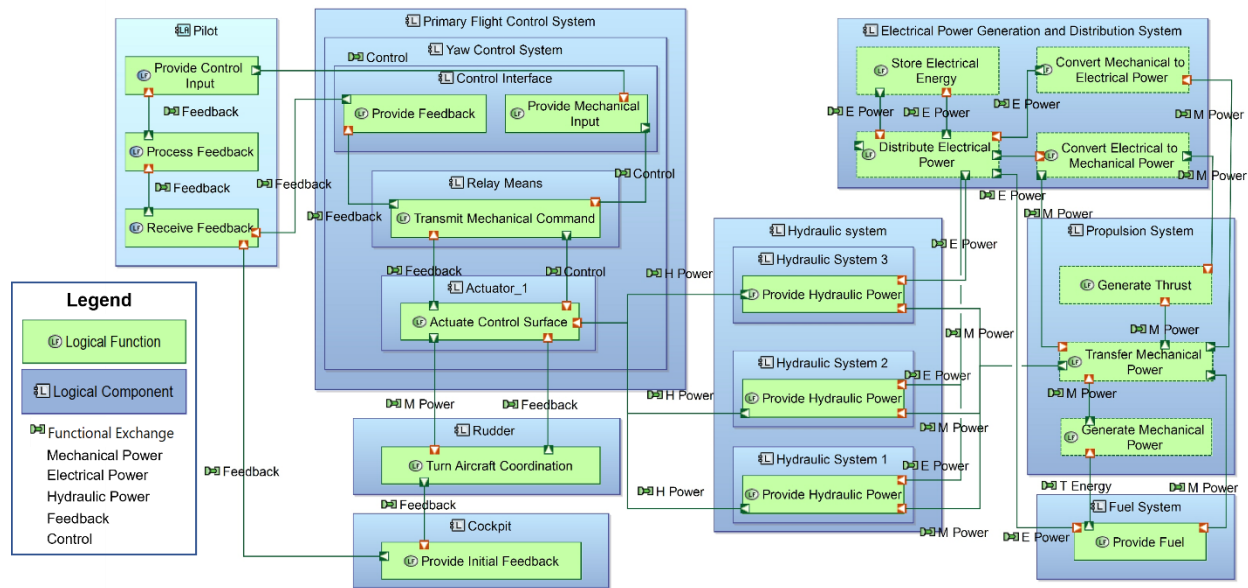


Figure 4- 12: Sys-L1 for FCS yaw control and interfaces to HE systems in Capella

As the pilot has a direct influence on the yaw control system, the Sys-L1 model shows the pilot as the logical actor of the system, depicted in a light blue box. The model also highlights important interactions, such as control and feedback chains, to help understand the relationships between various components and their effects on the overall system.

More detailed models of the yaw control system, including Sys-L2 and Sys-P2, are presented in Appendix. These diagrams contain the redundancy of all components and physical implementation for each logical behavior.

4.2 Test-Case 2: Model-based FHA for the Landing Gear Braking System

This section outlines the implementation of the proposed method using a case study of an aircraft wheel braking system, which was adapted from the SAE ARP4761 [24] example for the system FHA. The model-based FHA activities within Capella and enabling safety assessment are demonstrated through all proposed steps. As stated earlier, the initial step in the FHA process involves identifying functions at the aircraft and system levels. Figure 4-12 exhibits the decomposition of aircraft-level functions in the SA modeling layer within the scope of AFHA. Meanwhile, the proposed functions related to the landing gear and braking systems, as well as interface functions to the braking system functions, are highlighted in the LA modeling level within the scope of SFHA. The allocation of functions to systems is a crucial input to PASA, aiming to identify the system(s) that will implement the respective function(s).

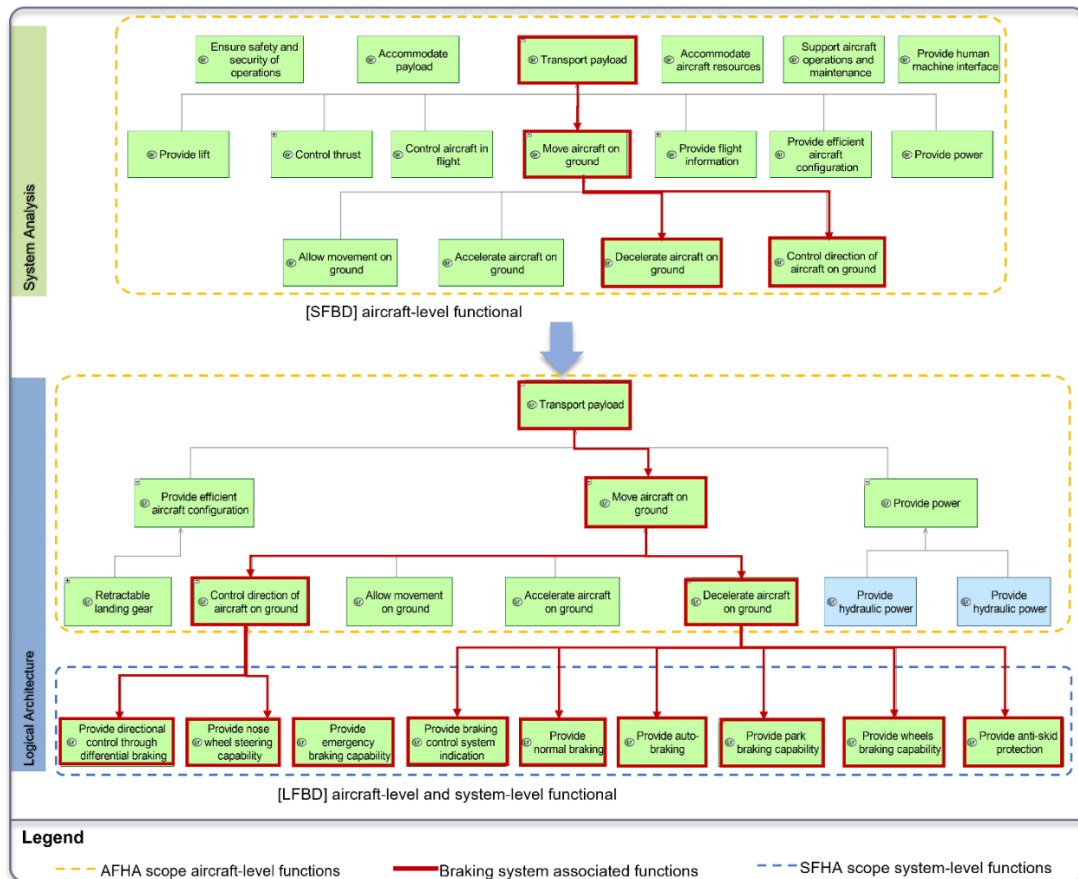


Figure 4- 13: Integration of AFHA and SFHA scopes functional decomposition in [SFBD] and [LFBD] diagrams in Capella

Using the system architecture diagram in the S0 model, top-level aircraft functions are allocated to the system of interest, aircraft, while low-level aircraft sub-functions are implemented using the Logical architecture diagram in the L0 model for the aircraft and its systems. Figure 4-14 (a) shows the system architecture diagram [SAB] in the S0 model and its allocated functions, while Fig. 4-14 (b) shows the logical architecture diagram [LAB], which includes the aircraft systems and the allocated sub-functions. This level of activity is necessary to organize aircraft functions and assign them to systems based on their appropriate grouping [23].

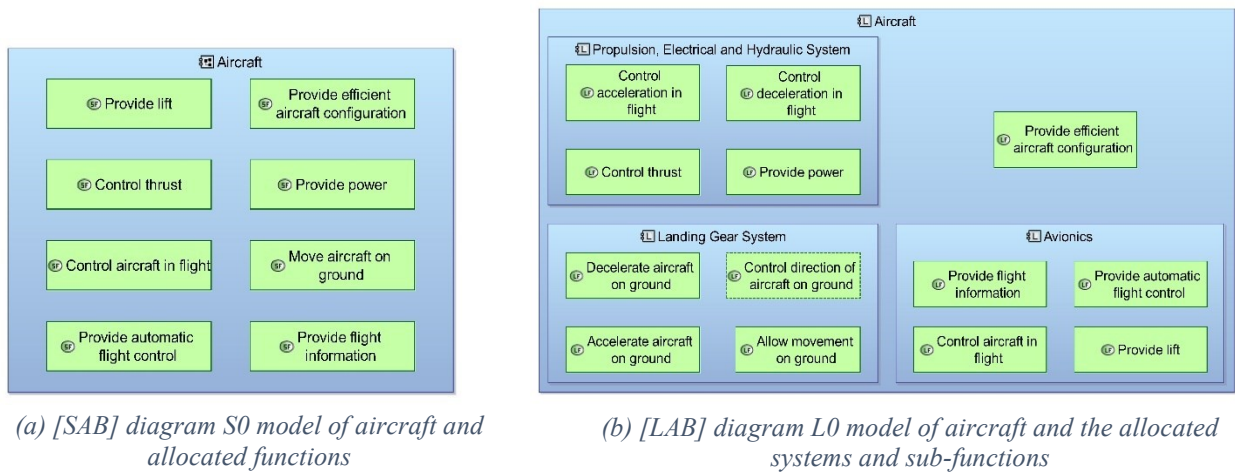


Figure 4- 14: Allocation of aircraft-level functions to the system(s) in Capella at SA and LA modeling levels

The allocation of functions to systems and the development of the system architecture are crucial steps for the SFHA. These activities are initiated at the S0 and L0 models in Capella and are further developed and refined at the L2 model. The output of this model is a complete system architecture, including all relevant details, such as component redundancy and interface elaborations [95]. These detailed models are valuable for PSSA activities and for determining a system's architectural and safety requirements [23].

Figure 4-15 illustrates the system architecture for the landing gear braking system at the L2 model. The wheel braking system, the system of interest, interfaces with the hydraulic system, electrical system, and pilot. The functions in the architecture are styled differently according to the most severe classification of their associated failure conditions in all flight phases using the color-coding feature in PMVT. As an example, the color coding is applied to the "Provide park braking

capability," "Provide normal braking," and "Provide auto-braking" functions. Green represents minor, yellow as major, orange as hazardous, and red as catastrophic failure conditions. During the taxiing phase, the failure condition of "loss of park braking mode capability" classifies "Provide park braking capability" as minor. However, normal braking and auto-braking are both classified as red since uncommanded braking during take-off is typically catastrophic.

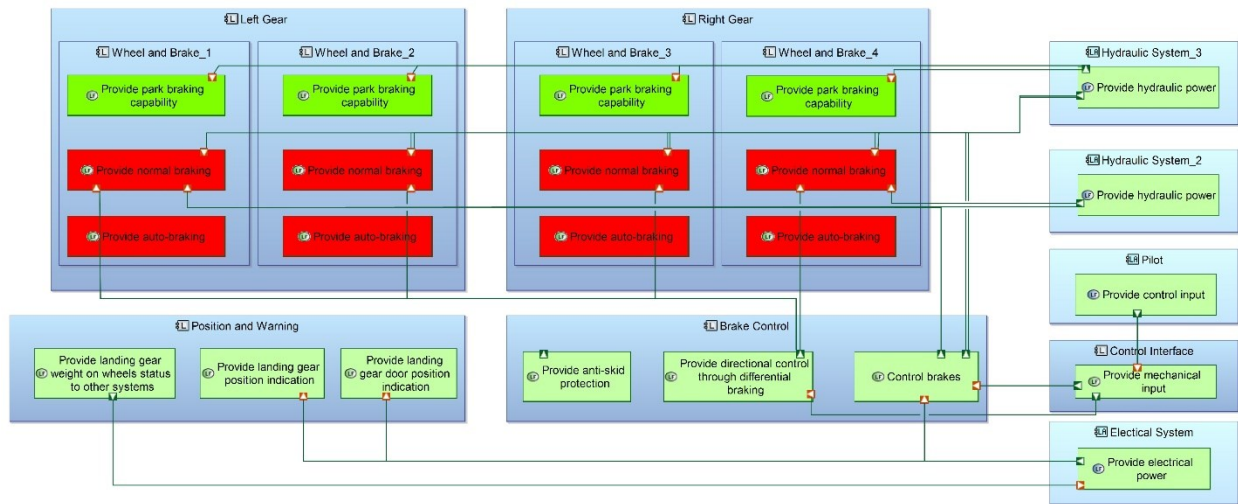


Figure 4- 15: Allocation of system-level functions to the braking system with color-coding at [LAB] logical architecture diagram L2 model in Capella

The proposed methodology for system safety analysis (SFHA) includes the use of functional chain diagrams at the system architecture (SA) and logical architecture (LA) levels to identify functional failure relationships and negative dataflows. These diagrams typically show functional relationships for accomplishing a mission, but in this methodology, they are used to represent failure relationships. The analysis is performed using the [SDFB] diagram in Capella to demonstrate failure relationships at the aircraft-level functions based on their properties, while the [LDFB] diagram represents failure effects relationships for system-level functions. Figure 4-16 provides an example of a functional failure chain diagram for the brakes system, showing failure relationships for both aircraft-level and system-level functions. This approach enables visualization of the potential chain of cause-and-effect failure relationships, supporting the safety assessment process.

After identifying the failure conditions of a function and specifying the impacted functions in the properties, it becomes possible to define the failure effect relationship between the failed and impacted functions. The functional failure diagram can provide valuable support to the safety

assessment process by allowing for the visualization of potential chains of cause-and-effect failure relationships. It is important to note, however, that this approach does not provide a systematic identification of failure events or combinations of events that may lead to a failure condition, which would typically fall under the scope of a Fault Tree Analysis. The concept of cause-and-effect can be described as the failure of one function, potentially leading to the failure of another function. When a chain of cause and effect occurs over a series of different functions, this is known as cascading or propagation effects.

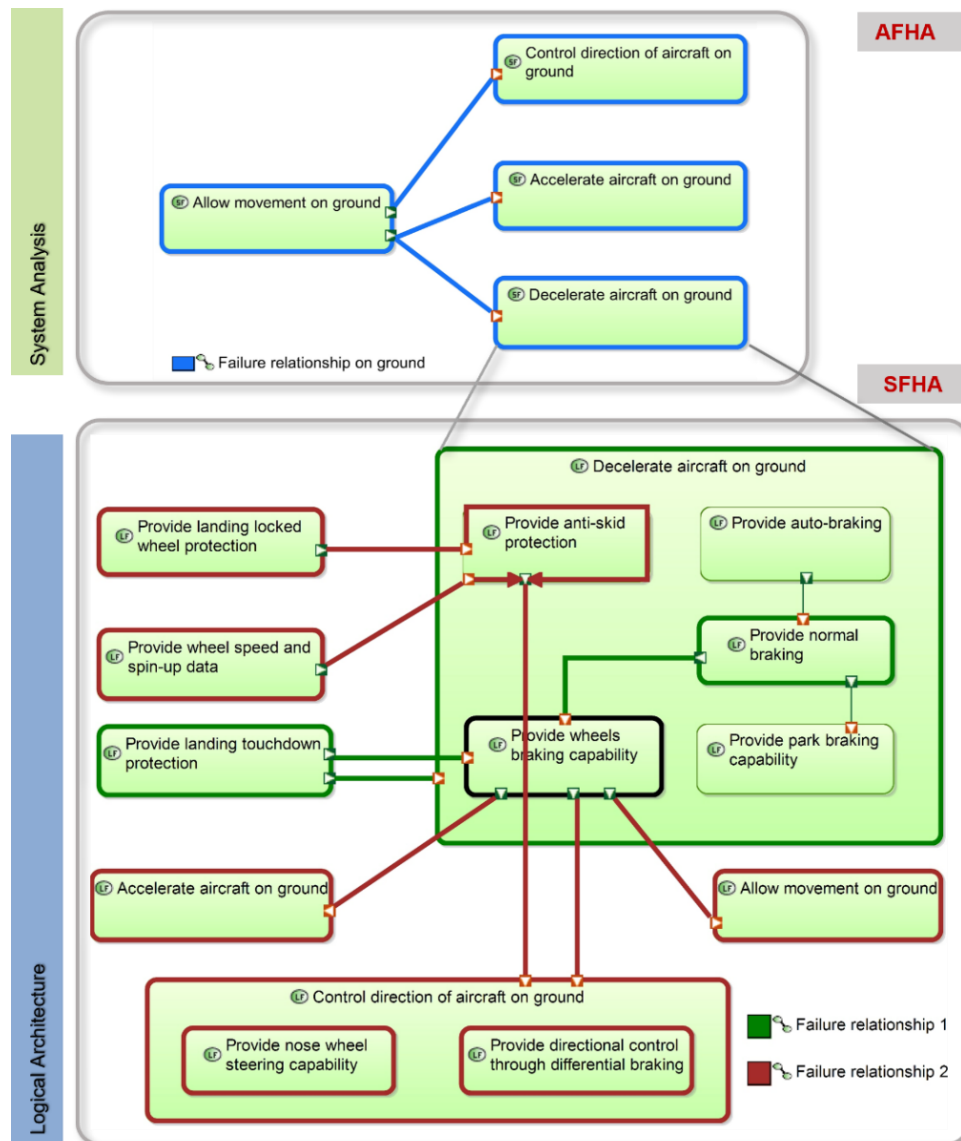


Figure 4-16: Functional failure effect relationships related to the brakes system functions using [SDFB] and [LFBD] Functional chain diagrams in Capella

The failure relationship chain depicted in Figure 4-16 can be explained by following the failure conditions associated with a function and its impacted functions. To further illustrate this relationship, an example is provided in Figure 4-17, which shows the failure relationship for the ground phase within the scope of AFHA.

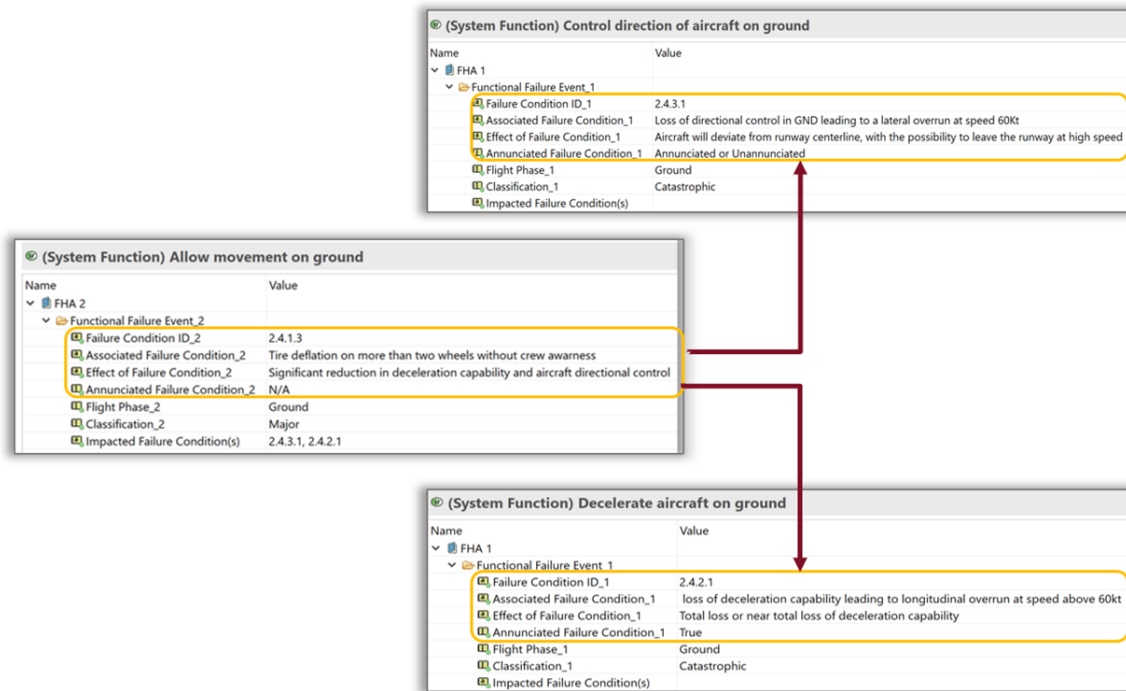


Figure 4- 17: Safety properties allocation to the functions evident the failure relationship on ground

In a similar vein, the [FS] diagram offers an alternative approach to identifying potential failure effects. In the scenario diagram Figure 4-18, system-level functions are depicted as green boxes with vertical lines indicating their execution and operation time. Here, the execution boxes as functional failure occurrences. The exchanged arrows between functions, representing failure relationships, allow different types of failure events to be identified. For example, "one-to-many" (a single function with several outputs) would suggest a resource function, and it is important to determine whether failure effects cover all the functions that may be impacted. Conversely, "many-to-one" (a single function with several inputs) would suggest an end-consumer or concentrator function, and it is essential to understand the impact of each input on failure conditions and their severity.

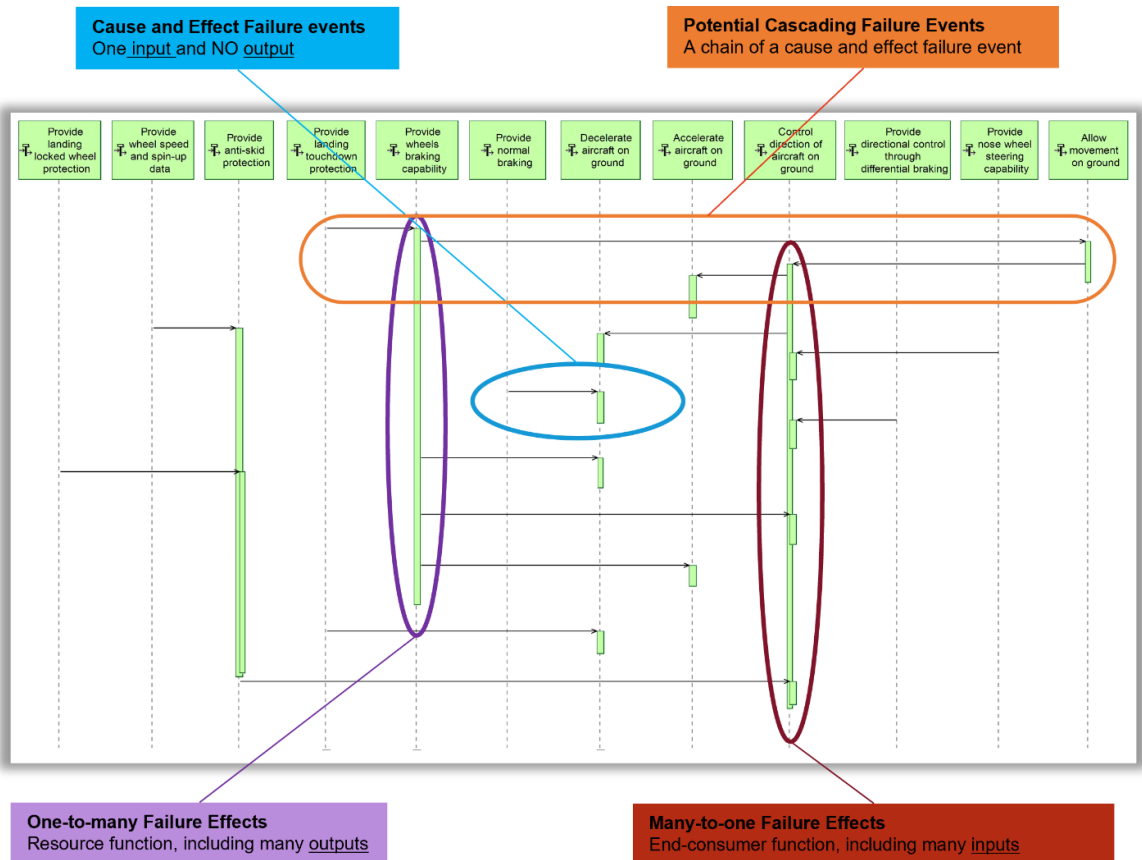


Figure 4- 18: Functional failure effect event and their relationships analysis potential using the functional scenarios [FS] diagram in Capella

The safety assessment process requires an analysis of failure effects and their severity across various flight and operational phases. To achieve this, the potential use of the modes and states diagram, or [MSM], in Capella, which is a transversal model, is investigated. As shown in Figure 4-19, the MSM diagram identifies the relevant flight phases and the functions expected to operate during each phase. By considering all relevant flight phases, the MSM diagram helps ensure that the failure conditions are complete. Additionally, the MSM diagram provides a graphical representation of the transitions between flight phases and any events that may disrupt the typical sequence of flight phases.

Figure 4-19 illustrates an instance of the Rejected Take-Off (RTO) scenario as a typical failure scenario. In order to model this scenario, the modes and states diagram is employed. The diagram illustrates the “Take-Off” phase, which can lead to two scenarios. The first scenario is the normal operation, which proceeds to the “Climb” mode and the succeeding phases until “taxiing” on the ground. However, if there is an event that requires the abortion of take-off, the aircraft enters the

“Rejected Take-Off (RTO)” mode. For instance, the function “Provide normal braking” is expected to be active during landing and RTO. Therefore, it is necessary to assess the “Loss of normal braking” in these two situations. On the other hand, this function is not expected to be active during take-off and taxi, and therefore, the “uncommanded normal braking” needs to be evaluated during these phases.

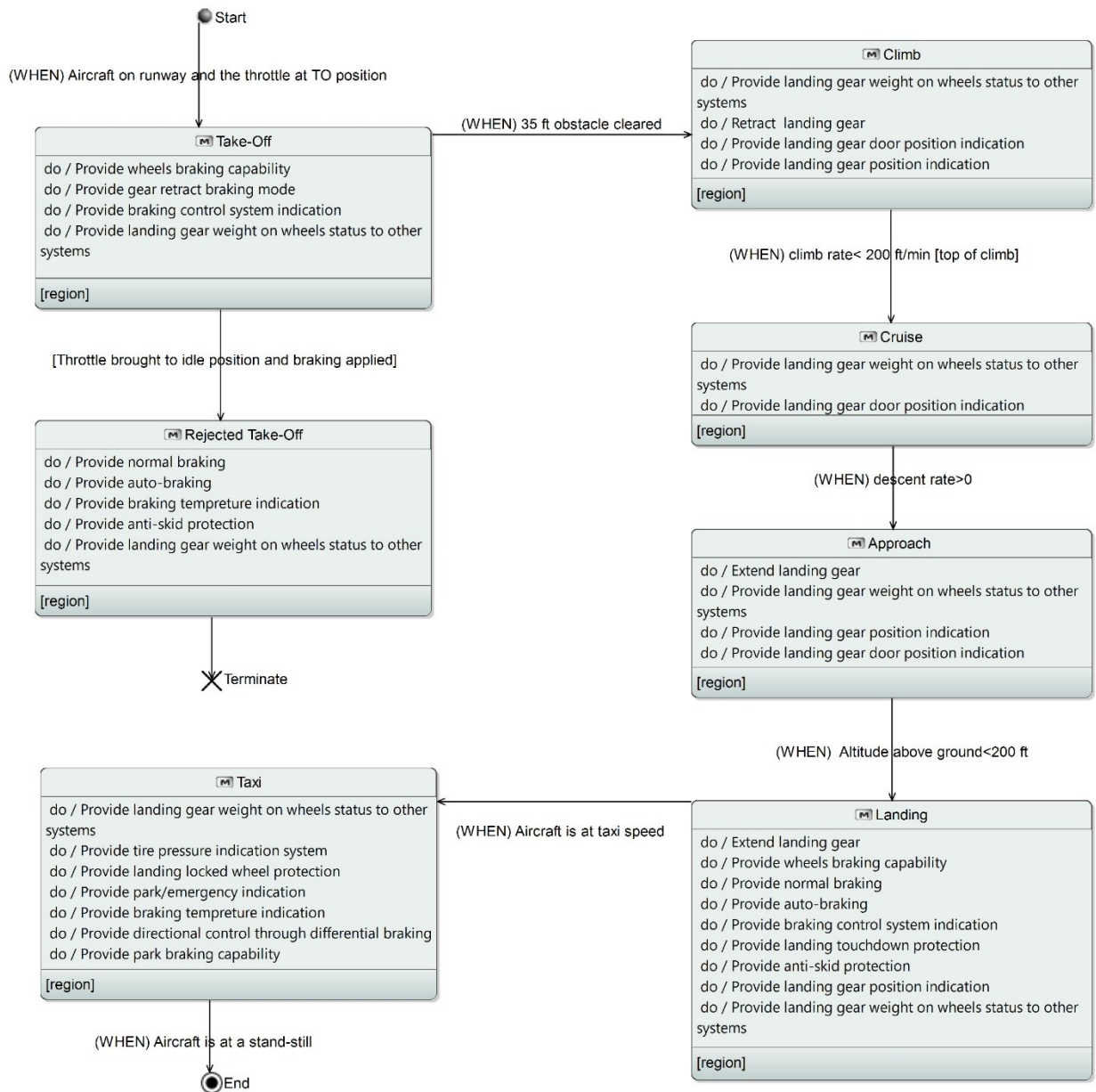


Figure 4- 19: Flight phases and expected operating functions during a normal operation flight and rejected take-off in [MSM] diagram in Capella,

The modes and states diagram can offer several benefits in terms of identifying critical flight phases and potential failure conditions. It also promotes a standardized definition of flight phases between the FHA and system architecting. The impact and criticality of a functional failure can be evaluated systematically by extracting all the flight phases in which a specific function is active. This approach can eventually lead to the creation of a comprehensive report documenting the analysis of the overall model.

4.3 Summary

This chapter presented two test cases that implement the framework developed in Chapter 3:

Test-case 1 is the HE architecture of the hybridized Do-228, built using the developed modeling elements at increasing levels of granularity. The S0 representation creates a simplified generic architecture of all types of HE systems. The use of generic elements ensures that the resulting diagrams are reusable. Similarly, L0 is generic but is tailored for different configurations of HE aircraft. For unconventional system architectures, L1 is suggested as middle-stage modeling in this framework to show the technological choices and key component redundancy. Also, it shows the interface system to highlight the connections of HE functions to other aircraft functions. However, L2 representation allows a profound view of the interaction between functions and their logical components while addressing all component redundancy as required for safety analysis. Physical implementations for logical behaviors are presented in the P2 model for all interface systems, such as yaw control, hydraulic, and fuel systems. Moreover, the sys-L1 model illustrates system-level functions with deeper insight into the yaw control system. System-level model elements are built based on Challenger 605 to simplify the architecture.

In test-case 2, the model-based FHA activities in Capella are tested for the brake system of landing gears. A customized safety viewpoint is created using the PVMT tool, and several diagrams in Capella are explored and utilized to support safety practices at an early stage of system development. Although test-case 2 adequately covers all the relevant aspects of FHA, there are still some challenges with the proposed method. One is navigating the complexity of the model-based specification with multiple layers of elaboration. Another is the manual process of

identifying failure effects due to Capella limitations. Future research will aim to address these challenges by automating the process and expanding the work to FTA and FMEA levels.

5 Conclusion and Future Work

This chapter summarizes the key contributions and implications of this thesis and a discussion thereof, subsequently outlining potential avenues for future exploration. It aims to synthesize the development within the context of aircraft system architecture early in the design stages, emphasizing the value added by this research and highlighting opportunities for future progression in this critical field of study.

5.1 Summary of Contributions and Discussion

An MBSE modeling approach for architecting system specifications in early design stages is developed using ARCADIA/Capella tool in this thesis. The proposed framework presents multi-level architecture templates, allowing unconventional aircraft architecture such as HE to be modeled from aircraft-level functions to physical implementations with different granularity levels. This can help in system architecting process by managing the complexity and variability of the architectures. Furthermore, it is explored how the MBSE specification model can be enhanced to ease the link to other early design activities, such as MDAO and safety assessment. This connection is necessary to evaluate architectures using system sizing, performance, and to enable optimization studies. Furthermore, the evaluation of architecture also provides information to enrich the MBSE specification. A framework is also developed to conduct early safety assessment activities such as FHA, which enables the identification and improves visibility of key functional failures and their effects on the system and at aircraft-level. This early enrichment of the system model with safety specific information, serves as an initial step towards MBSA. The work presents a method for transitioning the current document-based FHA process to an MBSE environment enabling a more comprehensive and, as a result, more effective safety evaluation in the early design phases.

The proposed system architecting framework enables aircraft system architectures to be represented and visualized during the conceptual design stage. Early representation of architectures can facilitate communication and collaboration among all project teams and stakeholders. This can help to ensure that everyone has a clear and consistent understanding of the system architecture and how it meets the requirements and standards. The use of generic elements and models in the representation of system architectures allows for a rapid exploration and evaluation of a large set of candidate architectures. This can help to identify the most promising architectures early in the design process and facilitate the evaluation of the system from various aspects. This expedited process enhances the efficiency of the overall system architecting process.

The utilization of the MBSE framework in aircraft development provides several advantages. One of the key benefits is the creation of reusable model elements that can be used in various stages of development, including the development of technology and interface specifications for system developers. This eliminates any vagueness in terms of system requirements and interface definitions, allowing the system integrator or aircraft manufacturer to specify the system that suits the aircraft's requirements. Moreover, the presented multi-level modeling approach in Capella gives users different views of the system's structures, including functional, logical, and physical in a variety of granularity levels. This can efficiently help to navigate across different detail levels of novel aircraft architectures. Also, external modules and add-ons to Capella can provide different "Viewpoints" to enrich the models in various domains to support model-based system analysis. These viewpoints can be customized according to the nature of the analysis and the type of desired output.

By leveraging the functional structures of the system architectures and customizing a safety viewpoint, model-based FHA activities can be integrated within the Capella. Using the MBSE framework in the FHA process can enhance traditional tables and documents with relationship diagrams and provide deeper insights for both the system and safety engineer and allow for easier iterations between them. Moreover, it enriches the model definition by providing a comprehensive view of the relevant constraints, including how certain failure conditions may be interrelated due to the inherent connection between system functions. The PVMT plugin to Capella was utilized to generate new properties linked to functions, which facilitate subsequent safety assessment tasks.

The presented method of linking MBSE and MDAO emphasizes the iterative process of systems architecting and the recent developments made in automating MDAO workflow specification by offering: a means for adding input properties to elements in the Capella MBSE and a means for extracting these properties to supply system-level MDAO tools. This can be an initial concept of integrating with the established MDAO specification and visualization environment developed by the AGILE 4.0 project.

The multi-level MBSE specification models are defined for a generic HE aircraft and its three common configurations: series, parallel, and partial series-parallel. The test case includes the propulsion system and electrical system as primary power generation, APU and mounted accessory gearbox as part of the secondary power generation, and hydraulic, fuel, and yaw control systems as power consuming. This work was performed as part of the collaborative research project in the frame of AGILE4.0 [26] and for the industrial partner Bombardier Inc.

Challenges

The proposed representation framework for systems architecture in the Capella MBSE tool has been found to have several drawbacks. While it meets the criteria of clarity, modularity, and traceability, it becomes difficult to read when the number of elements represented increases. This is due to the overlap between components and the resulting decrease in diagram readability. Moreover, when diagrams become larger, they are hard to visualize within a single view. Another limitation is the inability to zoom into diagrams, which can be an issue when a system component has an encapsulated detail representation within it. In the Capella tool, this feature is only supported in a separate model or diagram, and there is no capability to access the encapsulated representation directly from the main diagram.

The process of creating, sorting, and arranging the elements in Capella is time-consuming, especially when modeling complex system(s) in different granularity levels. At detail level 2, the system architecture representations consist of many elements, making it difficult to read the individual components and exchanges. This type of representation is effective in capturing system exchanges and serving as a comprehensive architecture specification for suppliers and manufacturers. It is not ideal for visualization and demonstration to other development teams. Furthermore, it is not possible to directly illustrate the location of system components with respect

to the aircraft. Overlaying Capella system architecture diagrams with aircraft drawings is challenging and time-consuming, providing no additional clarity.

The presented method of integrating MBSE and FHA in this thesis encompasses all the important aspects to formalize model-based FHA. Nevertheless, there are some challenges and limitations associated with this approach. Firstly, the model-based system specification with multi-levels of elaborations is sufficiently complex, making it difficult to navigate through the modeling levels. As the model-based FHA activities are performed at aircraft and system levels, transferring and exchanging the model specification between these levels within Capella requires higher levels of expertise. This can be a drawback as the safety engineers need a basic knowledge of MBSE to execute the FHA steps in Capella. Moreover, the current process of identifying and adding the failure effects to functions is performed manually. That is due to the Capella limitations in importing and exporting the element's properties and the inability of property inheritance between functions.

5.2 Future Work

One potential avenue for future work could be to investigate methods for improving the readability of large and complex system architecture representations in Capella. This could involve exploring alternative visualization techniques or developing tools for zooming in and out of encapsulated diagrams within a single view. Additionally, further research could focus on automating the process of diagram generation in Capella to improve efficiency and enable the exploration of a larger design space.

Further work could focus on developing a link between architecture specification and 3D visualization to improve system integrations and more accurately estimate system metrics such as weight and wiring length in the conceptual design phase. This can be achieved by generating parametric visualization of aircraft systems within a 3D modeling environment using physical architecture specifications.

Another aspect that requires further investigation is the link between MBSE and MDAO, focusing on improving automation and increasing flexibility in the workflow based on system architecture specifications. The ability to perform automated analysis on a larger design space will also be a

key area of investigation. In addition, researchers will explore the modeling levels required in MBSE for optimal MDAO workflow integration.

Moreover, for developing safety considerations of system architecture, more automated methods for failure affect recognition need to be explored to increase traceability and efficiency in the FHA process. The work can be extended beyond the FHA to the FTA and FMEA levels. This will allow for a more comprehensive safety assessment process in the early design stages.

The integration of these features will enable the exploration of aircraft system architectures further into the conceptual design phase, leading to an efficient aircraft development process. By addressing these areas, it is hoped that MBSE can become an even more powerful tool for optimizing complex system designs

List of Publications

- N. Tabesh, A. Jeyaraj, S. Liscouët-Hanke, and A. Tamayo, “Integration of the Functional Hazard Assessment within a Model-based System Engineering Framework,” in *AIAA SciTech Forum 2023*, 2023, pp. 1–16. doi: 10.2514/6.2023-1116.

Presented virtually at American Institute of Aeronautics and Astronautic (AIAA) Science and Technology Forum and Exposition National Harbor, MD, USA, February 2023.

- A. Jeyaraj, N. Tabesh, and S. Liscouët-Hanke, “Connecting Model-based Systems Engineering and Multidisciplinary Design Analysis and Optimization for Aircraft Systems Architecting – A case study within the AGILE4.0 project,” *AIAA Aviat. Aeronaut. Forum Expo. AIAA Aviat. Forum 2021*, 2021, doi: 10.2514/6.2021-3077.

Bibliography

- [1] A. Macintosh and L. Wallace, “International aviation emissions to 2025: Can emissions be stabilised without restricting demand?,” *Energy Policy*, vol. 37, no. 1, pp. 264–273, Jan. 2009, doi: 10.1016/J.ENPOL.2008.08.029.
- [2] S. Gössling and A. Humpe, “The global scale, distribution and growth of aviation: Implications for climate change,” *Glob. Environ. Chang.*, vol. 65, Nov. 2020, doi: 10.1016/j.gloenvcha.2020.102194.
- [3] “Aviation – CO2 Emission Analysis - IEA.” <https://www.iea.org/reports/aviation> (accessed Mar. 28, 2023).
- [4] Le Quéré, R.B. Jackson, M.W. Jones et al. “Temporary reduction in daily global CO2 emissions during the COVID-19 forced confinement.” *Nat. Clim. Chang.* 10, 647–653 (2020). <https://doi.org/10.1038/s41558-020-0797->
- [5] European Commission, Directorate-General for Mobility and Transport, Directorate-General for Research and Innovation, Flightpath 2050 – Europe’s vision for aviation : maintaining global leadership and serving society’s needs. Publications Office; 2012. doi/10.2777/15458
- [6] S. Bruner, S. Baber, C. Harris, N. Caldwell, P. Keding, K. Rahrig, L. Pho, and R. Wlezian, “NASA N + 3 Subsonic Fixed Wing Silent Efficient Low-Emissions Commercial Transport (SELECT) Vehicle Study Revision A. ” No. NASA Technical Report 2010-216798, 2010.
- [7] S. Gradel, B. Aigner, and E. Stumpf, “Model-based safety assessment for conceptual aircraft systems design,” *CEAS Aeronaut. J.*, Nov. 2021, doi: 10.1007/s13272-021-00562-2.
- [8] J. H. Bussemaker, L. Boggero, and P. D. Ciampa, “From System Architecting to System Design and Optimization: A Link Between MBSE and MDAO,” Jun. 2022.
- [9] N. Kodali Rao, “Influence of Parametric Modelling of Wing Subsystems on the Aircraft Design and Performance,” TU Delft, 2017. [Online]. Available: <https://www.agile-project.eu/cloud/index.php/s/3eOlyMgrFvO7F0t>
- [10] V. Mohan, A. K. Jeyaraj, and S. Liscouët-Hanke, “Systems Integration considerations for hybrid-electric commuter aircraft: a case study for the DO-228,” *AIAA SciTech Forum 2023*, 2023.
- [11] Y. XIE, A. SAVVARISAL, A. TSOURDOS, D. ZHANG, and J. GU, “Review of hybrid electric powered aircraft, its conceptual design and energy management methodologies,” *Chinese J. Aeronaut.*, vol. 34, no. 4, pp. 432–450, 2021, doi: 10.1016/j.cja.2020.07.017.
- [12] B. Litherland, N. Nicholas Borer, N. Nikolas Zawodny, and N. Zachary Frederick, “X-57 ‘Maxwell’ High-Lift Propeller Test for Improved Thrust Measurements and Slipstream Velocities.” 2022. Accessed: Mar. 28, 2023. [Online]. Available: <https://ntrs.nasa.gov/citations/20210017259>
- [13] “UTC’s Project 804 hybrid-electric demonstrator may increase regional jet efficiency by 30 percent”. [Online]. Available: <https://www.sae.org/news/2019/04/utc’s-project-804-hybrid-electric-demonstrator-may-increase-regional-jet-efficiency-by-30-percent> (accessed Mar. 28, 2023).

- [14] ZeroAvia, “First Practical Zero Emission Aviation Powertrain | USA & UK | ZeroAvia.”. [Online]. Available: <https://www.zeroavia.com/> (accessed Mar. 28, 2023).
- [15] M. K. Bradley, C. K. Droney, and T. J. Allen, “Subsonic Ultra Green Aircraft Research: : Phase II – Volume II – Hybrid Electric Design Exploration,” 2015.
- [16] F. Ralph H, J., Cheryl. Bowman, Amy, J., Rodger, D., and James L, “Overview of NASA Electrified Aircraft Propulsion Research for Large Subsonic Transports.” <https://ntrs.nasa.gov/citations/20170012222> (accessed May 26, 2022).
- [17] P. G. Juretzko, M. Immer, and J. Wildi, “Performance analysis of a hybrid-electric retrofit of a RUAG Dornier Do 228NG,” *CEAS Aeronaut. J.*, vol. 11, no. 1, pp. 263–275, Jan. 2020, doi: 10.1007/s13272-019-00420-2.
- [18] Hofmann, J.P.; Stumpf, E.; Weintraub, D.; Köhler, J.; Pham, D.; Schneider, M.; Dickhoff, J.; Burkhart, B.; Reiner, G.; Spiller, M.; et al. A comprehensive approach to the assessment of a hybrid electric powertrain for commuter aircraft. In *Proceedings of the AIAA Aviation 2019 Forum*, Dallas, TX, USA, 17–21; pp. 1–16, June 2019
- [19] Raymer, D. *Aircraft Design: A Conceptual Approach*. AIAA Education Series, Washington. DC, 1989.
- [20] Piperni, P.; DeBlois, A.; Henderson, R. “Development of a Multilevel Multidisciplinary Optimization Capability for an Industrial Environment. ” *AIAA J.* 2013, 51, 2335–2352.
- [21] J. S. Gray, J. T. Hwang, J. R. R. A. Martins, K. T. Moore, and B. A. Naylor, “OpenMDAO: an open-source framework for multidisciplinary design, analysis, and optimization,” *Struct. Multidiscip. Optim.*, vol. 59, no. 4, pp. 1075–1104, Apr. 2019, doi: 10.1007/s00158-019-02211-z.
- [22] P. D. Ciampa and B. Nagel, “The AGILE paradigm: The next generation of collaborative MDO,” in *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2017, doi: 10.2514/6.2017-4137.
- [23] SAE International, “ARP4754A: Development of Civil Aircraft and Systems,” 2011. doi: 10.1002/jmr.585.
- [24] SAE International, “Aerospace Recommended Practice 4761,” 1996.
- [25] I. Chakraborty and D. N. Mavris, “Heuristic Definition, Evaluation, and Impact Decomposition of Aircraft Subsystem Architectures,” *16th AIAA Aviation Technology, Integration, and Operations Conference*, Jun. 2016, doi: 10.2514/6.2016-3144.
- [26] “AGILE 4.0 – Towards cyber-physical collaborative aircraft development.” <https://www.agile4.eu/> (accessed Jun. 19, 2021).
- [27] IEEE, “ANSI/IEEE 1471-2000: IEEE Recommended Practice for Architectural Description of Software-Intensive Systems,” 2000.
- [28] A. K. Jeyaraj and S. Liscouët-Hanke, “A Safety-Focused System Architecting Framework for the Conceptual Design of Aircraft Systems,” *Aerospace*, vol. 9, no. 12, p. 791, Dec. 2022, doi: 10.3390/aerospace9120791.
- [29] S. Liscouët-Hanke, A. K. Jeyaraj. “A Model-Based Systems Engineering Approach for Efficient Flight Control System Architecture Variants Modelling in Conceptual Design.” In *Proceedings of the International Conference on Recent Advances in Aerospace Actuation Systems and Components*, Toulouse, France, 30 May–1 June 2018; pp. 34–41.

- [30] L. Boggero, P. D. Ciampa, and B. Nagel, “An MBSE architectural framework for the agile definition of complex system architectures,” in *AIAA AVIATION 2022 Forum*, Chicago, IL, USA, 27 June–1 July 2022. doi: 10.2514/6.2022-3720.
- [31] A. K. Jeyaraj. A Model-Based Systems Engineering Approach for Efficient System Architecture Representation in Conceptual Design: A Case Study for Flight Control Systems. Master Thesis, Concordia University, Montreal, QC, Canada, 2019.
- [32] Slow, G. Soremekun, H. Kim, and S. Spangelo, “Integrated model-based systems engineering (MBSE) applied to the Simulation of a CubeSat mission,” *2014 IEEE Aerospace Conference*. IEEE, Mar. 2014. doi: 10.1109/aero.2014.6836317.
- [33] S. Liscouët-Hanke, B. R. Mohan, P. Jeyarajan Nelson, C. Lavoie, and S. Dufresne, “Evaluating a Model-Based Systems Engineering approach for the conceptual design of advanced aircraft high-lift system architectures,” *Canadian Aeronautics and Space Institute AERO 2017*, 2017.
- [34] D. Huart and O. Olechowski, “Towards a Model-Based Systems Lifecycle: CPCS from design to operations,” in *International Workshop on Aircraft System Technologies*, 2017.
- [35] B. A. Morris, D. Harvey, K. P. Robinson, and S. C. Cook, “Issues in Conceptual Design and MBSE Successes: Insights from the Model-Based Conceptual Design Surveys,” *INCOSE Int. Symp.*, vol. 26, no. 1, pp. 269–282, Jul. 2016, doi: 10.1002/j.2334-5837.2016.00159.x.
- [36] A. Harish, J. Gladin, and D. Mavris, “Framework for design space exploration of novel propulsion system architectures,” *AIAA Scitech 2020 Forum*, vol. 1 PartF, 2020, doi: 10.2514/6.2020-1007.
- [37] S. Liscouët-Hanke, J. C. Maré, and S. Pufe, “Simulation framework for aircraft power system architecting,” *J. Aircr.*, vol. 46, no. 4, pp. 1375–1380, 2009, doi: 10.2514/1.41304.
- [38] T. Lammering, “Integration of Aircraft Systems into Conceptual Design Synthesis,” Ph.D. dissertation, RTWH Aachen University, Aachen, Germany, 2014.
- [39] J. H. Bussemaker, P. D. Ciampa, and B. Nagel, “System architecture design space exploration: An approach to modeling and optimization,” in *AIAA AVIATION 2020 FORUM*, 2020, vol. 1 PartF, pp. 1–22. doi: 10.2514/6.2020-3172.
- [40] A. K. Jeyaraj, J. Bussemaker, S. Liscouët-Hanke, L. Boggero. “Systems Architecting: A Practical Example of Design Space Modeling and Safety-Based Filtering within the AGILE4.0 Project.” In *Proceedings of the 33rd Congress of the International Council of the Aeronautical Sciences*, Stockholm, Sweden, 4–9 September 2022; ICAS: Stockholm, Sweden, 2022.
- [41] J. H. Bussemaker, P. D. Ciampa, and B. Nagel, “System architecture design space exploration: An approach to modeling and optimization,” *AIAA Aviation 2020 Forum*, vol. 1 PartF, pp. 1–22, 2020, doi: 10.2514/6.2020-3172.
- [42] P. Roques, “Systems architecture modeling with the Arcadia method : a practical guide to Capella.” 2018.
- [43] S. Liscouët-Hanke, “A model-based methodology for integrated preliminary sizing and analysis of aircraft power system architecture,” Ph.D. dissertation, University of Toulouse, Toulouse, France, Oct. 2008.
- [44] I. Chakraborty and D. N. Mavris, “Integrated assessment of aircraft and novel subsystem architectures in early design,” *J. Aircr.*, vol. 54, no. 4, pp. 1268–1282, 2017, doi: 10.2514/1.C033976.

- [45] M. Fioriti, L. Boggero, F. Tomasella, A. Mirzoyan, A. Isyanov, and P. S. Prakasha, "Propulsion and on-board system integration for regional, medium and long range jet with different level of systems electrification in the agile project," 2018. doi: 10.2514/6.2018-4847.
- [46] Boggero, L., Fioriti, M., Corpino, S., and Ciampa, P. D. "On-Board Systems Preliminary Sizing in an Overall Aircraft Design Environment." 17th AIAA Aviation Technology, Integration, and Operations Conference, Denver, Colorado, American Institute of Aeronautics and Astronautics Inc., Reston, Virginia, pp. 1–13, 2017, doi: 10.2514/6.2017-3065.
- [47] M. D. Guenov, A. Riaz, Y. H. Bile, A. Molina-Cristobal, and A. S. J. Heerden, "Computational framework for interactive architecting of complex systems," *Systems Engineering*, vol. 23, no. 3. Wiley, pp. 350–365, Feb. 17, 2020. doi: 10.1002/sys.21531.
- [48] I. Chakraborty and D. N. Mavris, "Assessing Impact of Epistemic and Technological Uncertainty on Aircraft Subsystem Architectures," 16th AIAA Aviation Technology, Integration, and Operations Conference., Jun. 10, 2016. doi: 10.2514/6.2016-3145.
- [49] S. Jimeno, A. Molina-Cristobal, A. Riaz, and M. Guenov, "Incorporating Safety in Early (Airframe) Systems Design and Assessment," *AIAA Scitech 2019 Forum*, Jan. 06, 2019. doi: 10.2514/6.2019-0553.
- [50] K. A. Odukoya, R. I. Whitfield, L. Hay, N. Harrison, and M. Robb, "An Architectural Description for the Application of Mbse in Complex Systems," *ISSE 2021 - 7th IEEE Int. Symp. Syst. Eng. Proc.*, 2021, doi: 10.1109/ISSE51541.2021.9582510.
- [51] "What is Model Based Systems Engineering? | Siemens Software." <https://www.plm.automation.siemens.com/global/en/our-story/glossary/what-is-model-based-systems-engineering/28573> (accessed May 03, 2023).
- [52] J. M. Borky and T. H. Bradley, *Effective Model-Based Systems Engineering*. Springer International Publishing, 2019. doi: 10.1007/978-3-319-95669-5.
- [53] N. A. Tepper, "Exploring the use of Model-Based Systems Engineering (MBSE) to develop Systems Architectures in Naval Ship Design," Master Thesis, Massachusetts Institute of Technology, 2010.
- [54] J. A. Estefan, "Survey of Model-Based Systems Engineering (MBSE) Methodologies," 2008. [Online]. Available: http://www.omgsysml.org/MBSE_Methodology_Survey_RevB.pdf
- [55] A. Maheshwari, "Industrial Adoption of Model-Based Systems Engineering: Challenges and Strategies," Master Thesis, Purdue University, 2015.
- [56] S. Gao, W. Cao, L. Fan, and J. Liu, "MBSE for satellite communication system architecting," *IEEE Access*, vol. 7, pp. 164051–164067, 2019, doi: 10.1109/ACCESS.2019.2952889.
- [57] D. Nichols and C. Lin, "Integrated Model-Centric Engineering: The Application of MBSE at JPL Through the Life Cycle," 2014. [Online]. Available: https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:06-iw14-mbse_workshop-application_of_mbse_at_jpl_through_the_lifecycle-nichols-lin-final.pdf
- [58] R. Malone, B. Friedland, J. Herrold, and D. Fogarty, "Insights from Large Scale Model Based Systems Engineering at Boeing," *INCOSE Int. Symp.*, vol. 26, no. 1, pp. 542–555, Jul. 2016, doi: 10.1002/j.2334-5837.2016.00177.x.
- [59] J. D’Ambrosio and G. Soremekun, "Systems engineering challenges and MBSE opportunities for automotive system design," in *2017 IEEE International Conference on Systems, Man, and*

- Cybernetics (SMC)*, 2017, pp. 2075–2080. doi: 10.1109/SMC.2017.8122925.
- [60] P. G. Mathew, S. Liscouet-Hanke, and Y. Le Masson, “Model-Based Systems Engineering Methodology for Implementing Networked Aircraft Control System on Integrated Modular Avionics - Environmental Control System Case Study,” *SAE Tech. Pap.*, vol. 2018-Novem, pp. 1–16, 2018, doi: 10.4271/2018-01-1943.
- [61] Z. C. Fisher, K. Daniel Cooksey, and D. Mavris, “A model-based systems engineering approach to design automation of SUAS,” in *2017 IEEE Aerospace Conference*, 2017, pp. 1–15. doi: 10.1109/AERO.2017.7943597.
- [62] C. Becker and T. Giese, “Application of Model Based Functional Specification Methods to Environmental Control Systems Engineering,” *SAE Int. J. Aerosp.*, vol. 4, no. 2, pp. 637–651, 2011, doi: 10.4271/2011-01-2504.
- [63] S. Liscouet-Hanke, H. Jahanara, and J. L. Bauduin, “A Model-Based Systems Engineering Approach for the Efficient Specification of Test Rig Architectures for Flight Control Computers,” *IEEE Syst. J.*, vol. 14, no. 4, pp. 5441–5450, Dec. 2020, doi: 10.1109/JSYST.2020.2970545.
- [64] J. H. Bussemaker and P. D. Ciampa, “MBSE in Architecture Design Space Exploration,” *Handb. Model. Syst. Eng.*, pp. 1–41, 2022, doi: 10.1007/978-3-030-27486-3_36-1.
- [65] F. Torrigiani et al., “An MBSE Certification-Driven Design of a UAV MALE Configuration in the AGILE 4.0 Design Environment,” *AIAA AVIATION 2021 FORUM*, Jul. 28, 2021. doi: 10.2514/6.2021-3080.
- [66] “Types of Models - SEBoK.” Accessed: Mar. 29, 2023. [Online]. Available: https://www.sebokwiki.org/wiki/Types_of_Models#Model_Classification
- [67] “What is UML | Unified Modeling Language.” <http://www.uml.org/what-is-uml.htm> (accessed May 04, 2023).
- [68] “SysML Open Source Project: What is SysML? Who created it?” <https://sysml.org/> (accessed May 04, 2023).
- [69] S. Friedenthal, *A Practical Guide to SysML: The Systems Modeling Language*. Burlington: Elsevier/Morgan Kaufmann Publishers, 2008.
- [70] J.-L. Voirin, “Motivations, Background and Introduction to Arcadia,” in *Model-based System and Architecture Engineering with the Arcadia Method*, M. Voirin, Ed. Elsevier, 2018, pp. 3–14. doi: 10.1016/B978-1-78548-169-7.50001-9.
- [71] J.-L. Voirin, “Modelling Languages for Functional Analysis Put to the Test of Real Life,” in *Complex Systems Design & Management*, 2012. [Online]. Available: <https://link.springer.com/content/pdf/10.1007%2F978-3-642-34404-6.pdf>
- [72] P. George Mathew, “Model-Based Systems Engineering Methodology for Implementing Networked Aircraft Control System on Integrated Modular Avionics – Environmental Control System Case Study,” Master Thesis, Concordia University, Montreal, QC, Canada, 2019.
- [73] “Capella MBSE Tool - Arcadia.” [Online]. Available: <https://www.eclipse.org/capella/addons.html> (accessed Nov. 25, 2022).
- [74] F. Mhenni, J.-Y. Choley, N. Nguyen, and C. Frazza, “Flight Control System Modeling with SysML to Support Validation, Qualification and Certification,” in *IFAC-PapersOnLine*, Dec. 2016, vol. 49, pp. 453–458. doi: 10.1016/j.ifacol.2016.07.076.

- [75] J. C. Chaudemar and P. De Saqui-Sannes, “MBSE and MDAO for Early Validation of Design Decisions: A Bibliography Survey,” Apr. 2021. doi: 10.1109/SYSCON48628.2021.9447140.
- [76] O. Aiello, D. S. D. R. Kandel, J. C. Chaudemar, O. Poitou, and P. De Saqui-Sannes, “Populating MBSE Models from MDAO Analysis,” Sep. 2021. doi: 10.1109/ISSE51541.2021.9582519.
- [77] P. D. Ciampa, G. La Rocca, and B. Nagel, “A MBSE Approach to MDAO Systems for the Development of Complex Products,” *AIAA Aviation 2020 FORUM.*, Jun. 08, 2020. doi: 10.2514/6.2020-3150.
- [78] C. Habermehl, G. Höpfner, J. Berroth, S. Neumann, and G. Jacobs, “Optimization Workflows for Linking Model-Based Systems Engineering (MBSE) and Multidisciplinary Analysis and Optimization (MDAO),” *Applied Sciences*, vol. 12, no. 11. MDPI AG, p. 5316, May 24, 2022. doi: 10.3390/app12115316.
- [79] P. Schmollgruber et al., “Use of a Certification Constraints Module for Aircraft Design Activities,” *17th AIAA Aviation Technology, Integration, and Operations Conference*, Jun. 05, 2017. doi: 10.2514/6.2017-3762
- [80] M.-H. Bleu-Laine, M. V. Bendarkar, J. Xie, S. I. Briceno, and D. N. Mavris, “A Model-Based System Engineering Approach to Normal Category Airplane Airworthiness Certification,” *AIAA Aviation 2019 Forum*, Jun. 15, 2019. doi: 10.2514/6.2019-3344.
- [81] I. van Gent, G. La Rocca, and M. F. M. Hoogreef, “CMDOWS: a proposed new standard to store and exchange MDO systems,” *CEAS Aeronaut. J.*, vol. 9, no. 4, pp. 607–627, 2018, doi: 10.1007/s13272-018-0307-2.
- [82] M. Alder, E. Moerland, J. Jepsen, and B. Nagel, “Recent Advances in Establishing a Common Language for Aircraft Design with CPACS,” *Aerospace Europe Conference 2020, Jun. 2020*.
- [83] O. Lisagor, T. Kelly, and R. Niu, “Model-based safety assessment: Review of the discipline and its challenges,” in *ICRMS'2011 - Safety First, Reliability Primary: Proceedings of 2011 9th International Conference on Reliability, Maintainability and Safety*, Jun. 2011, pp. 625–632. doi: 10.1109/ICRMS.2011.5979344.
- [84] O. Lisagor, A. Mcdermid, and D. J. Pumfrey, “Towards a Practicable Process for Automated Safety Analysis,” 2006.
- [85] G. Girard et al., “Model based Safety Analysis using SysML with Automatic Generation of FTA and FMEA Artifacts,” *Proceedings of the 30th European Safety and Reliability Conference and 15th Probabilistic Safety Assessment and Management Conference*, 2020. doi: 10.3850/978-981-14-8593-0_4941-cd.
- [86] A. A. Abdellatif and F. Holzapfel, “Model Based Safety Analysis (MBSA) Tool for Avionics Systems Evaluation,” in *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, 2020, pp. 1–5. doi: 10.1109/DASC50938.2020.9256578.
- [87] T. Prosvirnova et al., “The AltaRica 3.0 project for model-based safety assessment,” in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2013, vol. 4, no. PART 1, pp. 127–132. doi: 10.3182/20130904-3-UK-4041.00028.
- [88] S. Maitrehenry, S. Metge, Y. Ait-Ameur, and P. Bieber, “Towards Model-Based Functional Hazard Assessment at Aircraft Level,” 2011.
- [89] M. Sango, F. Vallée, A.C. Vié, J.L. Voirin, X. Leroux, and V. Normand, “MBSE and MBSA with Capella and Safety Architect Tools,” *Complex Systems Design & Management*. Springer

- International Publishing, pp. 239–239, Dec. 09, 2016. doi: 10.1007/978-3-319-49103-5_22.
- [90] F. Bruno, M. Fioriti, G. Donelli, L. Boggero, P. D. Ciampa, and B. Nagel, “A Model-Based RAMS Estimation Methodology for Innovative Aircraft on-board Systems supporting MDO Applications,” *AIAA Aviation 2020 FORUM*, Jun. 08, 2020. doi: 10.2514/6.2020-3151.
- [91] E. Villhauer and B. Jenkins, “An Integrated Model-Based Approach to System Safety and Aircraft System Architecture Development,” *INCOSE Int. Symp.*, vol. 25, no. 1, pp. 1373–1387, Oct. 2015, doi: <https://doi.org/10.1002/j.2334-5837.2015.00136.x>.
- [92] S. Jimeno, A. Riaz, M. D. Guenov, and A. Molina-Cristobal, “Enabling interactive safety and performance trade-offs in early airframe systems design,” in *AIAA Scitech 2020 Forum*, Jan. 2020, vol. 1 PartF, pp. 1–16. doi: 10.2514/6.2020-0550.
- [93] Y. Jiang, N. Bai, H. Yang, H. Zhang, Z. Wang, and X. Liu, “MBSE-based functional hazard assessment of civil aircraft braking system,” *Proc. - 2020 5th Int. Conf. Mech. Control Comput. Eng. ICMCCE 2020*, pp. 460–464, Dec. 2020, doi: 10.1109/ICMCCE51767.2020.00107.
- [94] A. Jeyaraj, N. Tabesh, and S. Liscouët-Hanke, “Connecting Model-based Systems Engineering and Multidisciplinary Design Analysis and Optimization for Aircraft Systems Architecting – A case study within the AGILE4.0 project,” *AIAA Aviation 2021 FORUM*, 2021, doi: 10.2514/6.2021-3077.
- [95] N. Tabesh, A. K. Jeyaraj, A. Tamayo, and S. Liscouët-Hanke, “Integration of the Functional Hazard Assessment within a Model-based System Engineering Framework,” *AIAA SciTech 2023 FORUM*, Jan. 19, 2023. doi: 10.2514/6.2023-1116.
- [96] Eclipse Capella - Open Source MBSE Solution, “[HOW TO]Replicate model elements in Capella,” 2019. www.youtube.com/watch?v=h-ax61eVlxM (accessed Feb. 08, 2023).
- [97] Air Transport Association, “ATA 100.” [http://www.s-techent.com/ATA100.htm#\[61\]](http://www.s-techent.com/ATA100.htm#[61]) (accessed Feb. 13, 2023).
- [98] D. Seider, P. M. Fischer, M. Litz, A. Schreiber, and A. Gerndt, “Open source software framework for applications in aeronautics and space,” *2012 IEEE Aerospace Conference*, Mar. 2012. doi: 10.1109/aero.2012.6187340.
- [99] “Engineering Process Integration | Noesis Solutions.” <https://www.noessolutions.com/our-products/optimus/engineering-process-integration> (accessed Jun. 17, 2021).
- [100] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework 2.0*, 2nd ed. Addison-Wesley Professional, 2009.
- [101] I. van Gent, G. La Rocca, and L. L. Veldhuis, “Composing MDAO symphonies: graph-based generation and manipulation of large multidisciplinary systems,” *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Jun. 02, 2017. doi: 10.2514/6.2017-3663.
- [102] “KE-works.” [Online]. Available: <https://ke-chain.com> (accessed Jun. 22, 2021).
- [103] Chiesa, S., Fioriti, M., and Viol, N. "Methodology for an Integrated Definition of a System and Its Subsystems: The Case- Study of an Airplane and Its Subsystems." *Systems Engineering - Practice and Theory*, 2012. doi: 10.5772/34453.
- [104] S. Bonnet, J.-L. Voirin, D. Exertier, and V. Normand, “Modeling system modes, states, configurations with Arcadia and Capella: method and tool perspectives,” *INCOSE International*

- Symposium, vol. 27, no. 1. Wiley, pp. 548–562, Jul. 2017. doi: 10.1002/j.2334-5837.2017.00378.x
- [105] J. L. Voirin, “Model-based system and architecture engineering with the arcadia method,” *Model. Syst. Archit. Eng. with Arcadia Method*, pp. 1–368, Nov. 2017, doi: 10.1016/c2016-0-00862-8.
- [106] R. De Vries, M. Brown, and R. Vos, “Preliminary sizing method for hybrid-electric distributed-propulsion aircraft,” *J. Aircr.*, vol. 56, no. 6, pp. 2172–2188, Oct. 2019, doi: 10.2514/1.C035388.
- [107] “Commercial aircraft propulsion and energy systems research: Reducing global carbon emissions,” *Commer. Aircr. Propuls. Energy Syst. Res. Reducing Glob. Carbon Emiss.*, pp. 51–53, Sep. 2016, doi: 10.17226/23490.
- [108] P. Strathoff, M. Nuño, K.-U. Schröder, and E. Stumpf, “A Study on ‘Through-the-Road’-Parallel-Hybrid Powertrains for Small Aircraft with Distributed Electric Propulsion,” *AIAA Aviation 2019 FORUM*, Jun. 14, 2019. doi: 10.2514/6.2019-3677.
- [109] A. S. Gohardani, G. Doulgeris, and R. Singh, “Challenges of future aircraft propulsion: A review of distributed propulsion technology and its potential application for the all electric commercial aircraft,” *Prog. Aerosp. Sci.*, vol. 47, no. 5, pp. 369–391, Jul. 2011, doi: 10.1016/J.PAEROSCI.2010.09.001.
- [110] Bombardier Inc., “Bombardier Challenger 605 - Flight Control.” [Online]. Available: https://www.smartcockpit.com/docs/CL605-FLIGHT_CONTROLS.pdf (accessed Mar. 15, 2023).
- [111] Bombardier Inc., “Bombardier Challenger 605 - Hydraulic System.” [Online]. Available: https://www.smartcockpit.com/docs/CL605-HYDRAULIC_SYSTEM.pdf (accessed Mar. 15, 2023)
- [112] Bombardier Inc., “Bombardier Challenger 605 - Fuel System.” [Online]. Available: https://www.smartcockpit.com/docs/CL605-FUEL_SYSTEM.pdf (accessed Mar. 15, 2023)

Appendix

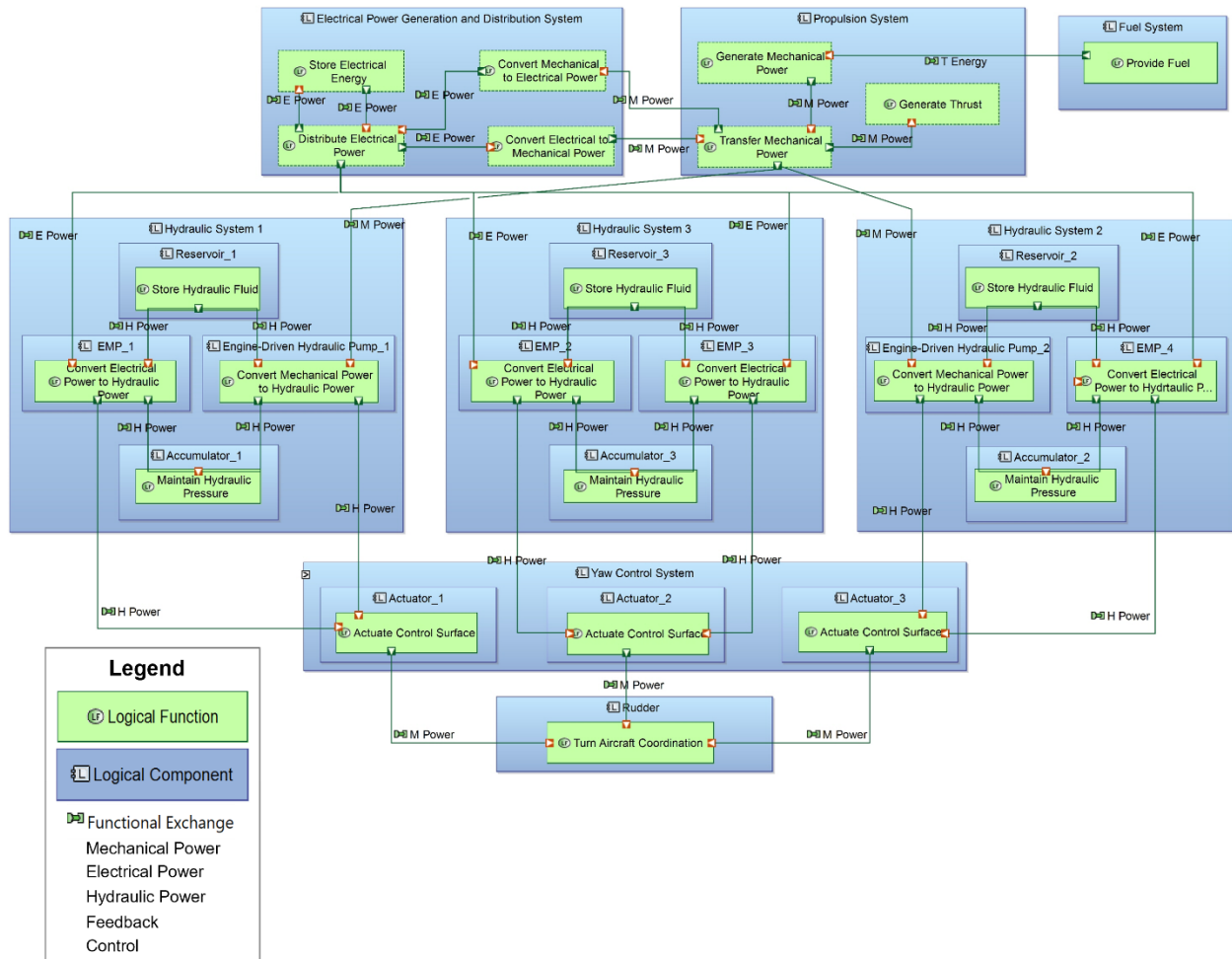


Figure A- 1: Sys-L1 for hydraulic system and interfaces to HE systems in Capella

Figure A-1 shows the Sys-L1 model for the hydraulic system. In the model, three hydraulic systems are defined and distinguished by numbers. Hydraulic system 1 and 2 has two types of hydraulic pump: Engine Driven Pump and Electrical Motor Pump, while two Electrical Motor Pump runs the hydraulic system 3. Each hydraulic system supplies one actuator in the yaw control system.

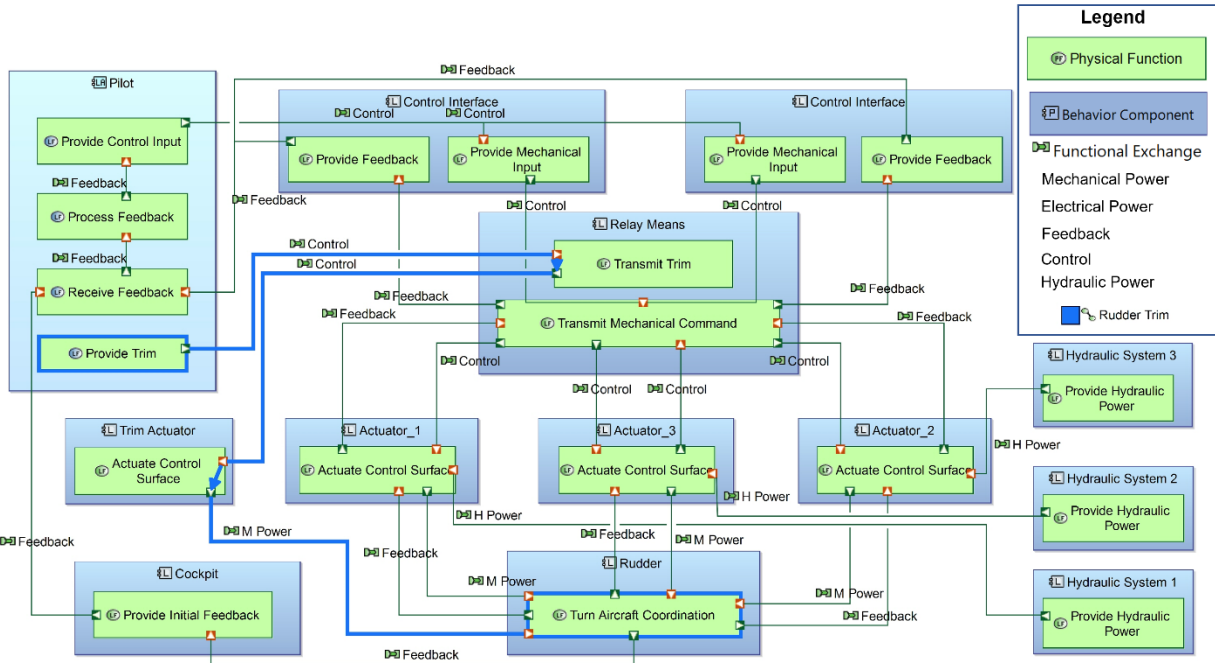


Figure A- 2: Sys-L2 for FCS yaw control and interfaces to HE systems in Capella

The Sys-L2 model for yaw control is illustrated in Figure A-2. Here, the detail view of the yaw control, such as control surfaces, relay means, rudder trim, and three actuators moving the rudder, are specified.

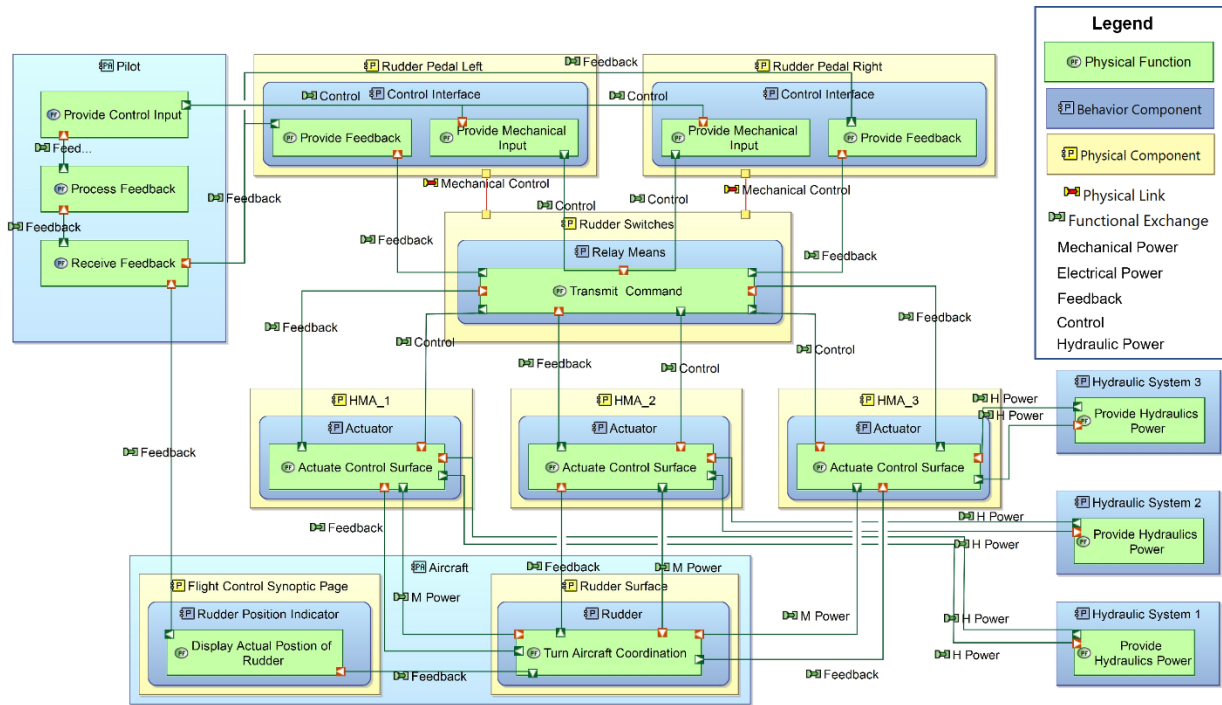


Figure A- 3: Sys-P2 for FCS yaw control and interfaces to HE systems in Capella

Figure A-3 shows the allocation of logical components to physical components in the model Sys-P2 for yaw control. For example, the physical component responsible for relay means is " Rudder Switches", and the Hydro mechanical Actuator is chosen for control surface actuation. The reference architecture for this model is Challenger 605. In order to simplify the architecture, the yaw damping system is neglected.

Table A-1: Propulsion and electrical systems nomenclature of model components in Capella

Modelling layers					
System Analysis	Logical Architecture		Physical Architecture		
System Function	Logical Function	Logical Component	Physical Function	Physical Component Behaviour	Physical Component Node
Propulsion System					
Generate Mechanical Power	Generate Mechanical Power	Engine/ Gas Turbine/ Internal Combustion Engine (generic type or sub-type)	Generate Mechanical Power	Engine/ Gas Turbine	Dual-spool turboprop
–	Transfer Mechanical Power	Accessory Gearbox	Transfer Mechanical Power	Accessory Gearbox	Accessory Gearbox
–	Reduce Shaft Speed	Propeller Speed Reduction Gearbox	Reduce Shaft Speed	Propeller Speed Reduction Gearbox	Propeller Speed Reduction Gearbox
–	Generate Support-Thrust	Supporting Propeller	Generate Thrust	Supporting Propeller	Electrically driven variable pitch
Generate Thrust	Generate Thrust	Propeller	Generate Thrust	Propeller	5-blade MTV-27
Electrical System					
Store Energy	Store Electrical Energy	Battery	Store Electrical Energy	Battery	Pouchbag cell (NMC) Battery
–	Convert Mechanical to Electrical Power	APU Generator	Convert Mechanical to Electrical Power	APU Generator	AC Generator
Convert Mechanical to Electrical Power	Convert Mechanical to Electrical Power	Generator	Convert Mechanical to Electrical Power	Generator	Switch Reluctance Machine AC Engine-Driven Generator
Convert Electrical to Mechanical Power	Convert Electrical to Mechanical Power	Motor	Convert Electrical to Mechanical Power	Motor	Switch Reluctance Machine
–	Convert Electrical to Mechanical Power	Wingtip Motor	Convert Electrical to Mechanical Power	Wingtip Motor	Permanent magnet synchronous motors
Convert Electrical Power	Convert DC to AC	Inverter	Convert DC to AC	Inverter	Inverter
Convert Electrical Power	Convert DC to DC	Converter	Convert DC to DC	Converter	Converter
–	Distribute Electrical Power	PDU	Distribute Electrical Power	PDU	DC Bus

Table A- 2: Hydraulic and fuel system nomenclature of models components in Capella

Modeling layers					
System Analysis	Logical Architecture		Physical Architecture		
System Function	Logical Function	Logical Component	Physical Function	Physical Component Behaviour	Physical Component Node
Hydraulic System					
-	-	EMP	Convert Electrical to Hydraulic Power	EMP	ACMP
-	-	EDHP	Convert Mechanical to Hydraulic Power	EDHP	EDP
-	-	Accumulator	Maintain Hydraulic Pressure	Accumulator	Nitrogen-Charged Cylinder
-	-	Reservoir	Store Hydraulic Fluid	Reservoir	Reservoir
Fuel System					
Store Energy	Store Fuel	Fuel Tank	Store Fuel	Fuel Tank	Main Tank - Right/ Left Wing
-	-	Fuel Tank	Store Fuel	Fuel Tank	Auxiliary Fuel Tank - Forward/Center/Aft
-	-	Fuel Tank	Store Fuel	Fuel Tank	Tail Tank - Cone - Left/Right Saddle
-	Transfer Fuel	Transfer Pump	Transfer Fuel	Transfer Pump	Ejector Pump
-	Deliver Fuel at proper Pressure	Feed Pump	Deliver Fuel at proper Pressure	Feed Pump	Engine-Driven Fuel Pump
-	Transfer Fuel	Feed Pump	Transfer Fuel	Feed Pump	DC Boost Pump
-	Transfer Fuel	Transfer Pump	Transfer Fuel	Transfer Pump	APU fuel Pump

Table A- 3: Primary flight control system nomenclature of model components in Capella

Modeling layers					
System Analysis	Logical Architecture		Physical Architecture		
System Function	Logical Function	Logical Component	Physical Function	Physical Component Behaviour	Physical Component Node
Primary Flight Control System – Yaw Control					
Turn Aircraft Coordination	Rudder	Rudder	Rudder	Turn Aircraft Coordination	Rudder
Transmit Mechanical Command	Relay Means	Relay Means	Control Cables	Transmit Mechanical Command	Relay Means
Transmit Trim	Relay Means	Relay Means	Trim Switch	Transmit Trim	Relay Means
Actuate Control Surface	Actuator	Actuator	HMA / EHSA - Upper/Middle/lower Rudder	Actuate Control Surface	Actuator
Actuate Control Surface	Trim Actuator	Trim Actuator	Trim Switch	Actuate Control Surface	Trim Actuator
Provide Initial Feedback	Cockpit	Cockpit	Flight Control Synoptic Page	Provide Initial Feedback	Cockpit
Provide Feedback	Control Interface	Control Interface	Pedals	Provide Feedback	Control Interface
Provide Mechanical Input	Control Interface	Control Interface	Pedals	Provide Mechanical Input	Control Interface

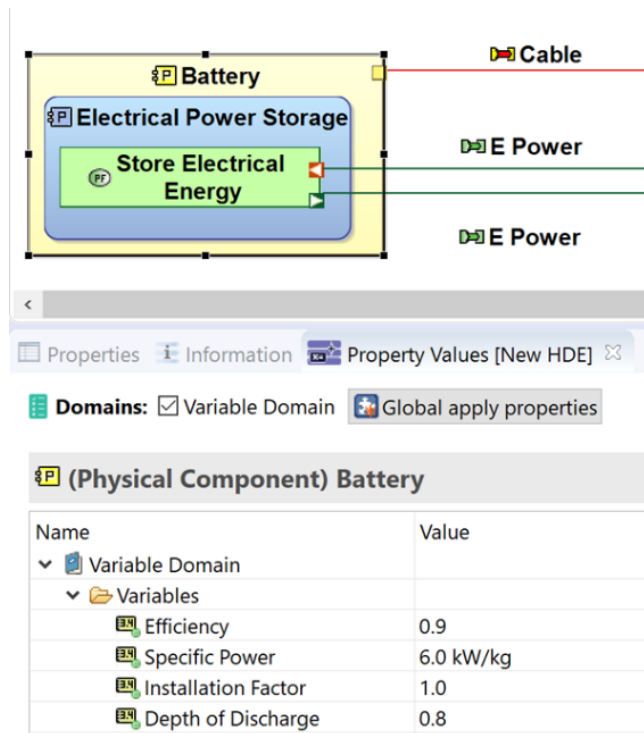


Figure A- 4: Allocation of properties to battery physical component in Capella using the PVMT add-on

Figure A-4 shows properties such as efficiency, specific power, installation factor, and depth of discharge applied to a physical component representing a battery. These properties can provide input to a battery sizing tool that will provide the battery's overall weight.

Table A- 4: Mapping between Capella schema and simplified schema tag names

Capella Schema Tag Name	Simplified Schema Tag Name
Root:org.polarsys.capella.core.data.capellamodeller:Project	Project
Root:ownedPropertyValuePkgs	Property Value Packages
Child:owned PropertyValueGroups	Property Value Groups
Child:ownedPropertyValues	Property Values
Root: ownedModelRoots	Models
Child: ownedArchitectures	Architectures
Child:ownedPhysicalComponentPackage	Physical Architecture
Child:ownedPhysicalComponents	Components
Child:ownedPropertyValueGroups	Property Value Groups
Child:ownedPropertyValues	Property Values

Each model artifact is assigned a unique alphanumeric sequence as an identifier within the schema, and links between elements such as functional chains and physical links reference these identifiers in specifying the source and target of each exchange. Although the schema is well structured, the size and tag names can make navigation cumbersome. In order to simplify the traversal of different tags for property extraction and the automation of the process, a simplified schema is created which contains only the tags that are accessed to extract model element properties. The mapping from the actual schema tags to a simplified version is shown in Table A-4, supplemented by an illustration in Figures A-5 and A-6.

The process of parsing and reading component parameter and data from Capella XML source code is as follow:

Step: 1: Access the Physical Components tag and check the Components tag against every name in the list of physical components.

Step 2: For every name that matches, read the name, id and appliedPropertyValue fields.

Step 3: Open the Property Value Groups tag and compare the id against the value stored in appliedPropertyValues from the previous step. For every match, read the Property Values tag for the id, name, and appliedPropertyValues fields.

Step 4: Open Property Value Packages and its child tag Property Values. There will be multiple child tags of the same name, so it is necessary to iterate through them and match the applied property values field. If a match does occur, then the name, description and s_id fields are read and stored.

Step 5: The tags from Step 3 (namely Property Values) are accessed once again and the value field is accessed and its contents are stored.

Step 6: The name and description field from Step 4 and the Value field from Step 5 are combined and provided as output. The description field from Step 4 stores the variable abbreviation as needed by any system-level tools. It is important that the matching variable names are set during the property assignment process.

This approach is used to extract the parameters listed in Table A-4 that are assigned to elements of the HE model- which include the name, variable name and value. This data is stored in an intermediary file, which in this case is an Excel sheet from where it is passed to a HE aircraft sizing and performance estimation tool. The results, which include, fuel weight and battery weight are written back to the Excel file and added to the Capella model through modification of the schema.

The steps required for adding properties to the Capella data model schema are as follows:

Step 1: Add another child Property Value Groups tag under the Property Value Packages tag. Add the name and id of the property category being written. It is imperative that the id be unique as it is referenced in subsequent tags.

Step 2: Add a child Property Value tag under the Property Value Group tag. Provide a name and unique id for the property being written.

Step 3: Add a child property value groups tag under the components tag of the physical component to which the property is being assigned. Provide a name, unique id and under the appliedPropertyValueGroups field, including the unique id referenced in step 2 prefaced with a ‘#’.

Step 4: Add a child Property Values tag under the Property Value Groups Tag of the selected physical component and include a unique id, name, an appliedPropertyValues field containing the nique id defined in step 3 prefaced with a ‘#’. Add a value field and enter the numeric or string value that is to be assigned to that property.

In the case of the HE system architecture, the weight of fuel and mass of battery are written back to the Capella schema and viewed in the Capella workbench. Duplicates are dealt with using the “s_id” field. If two physical components have the same name then the s_id values are compared.

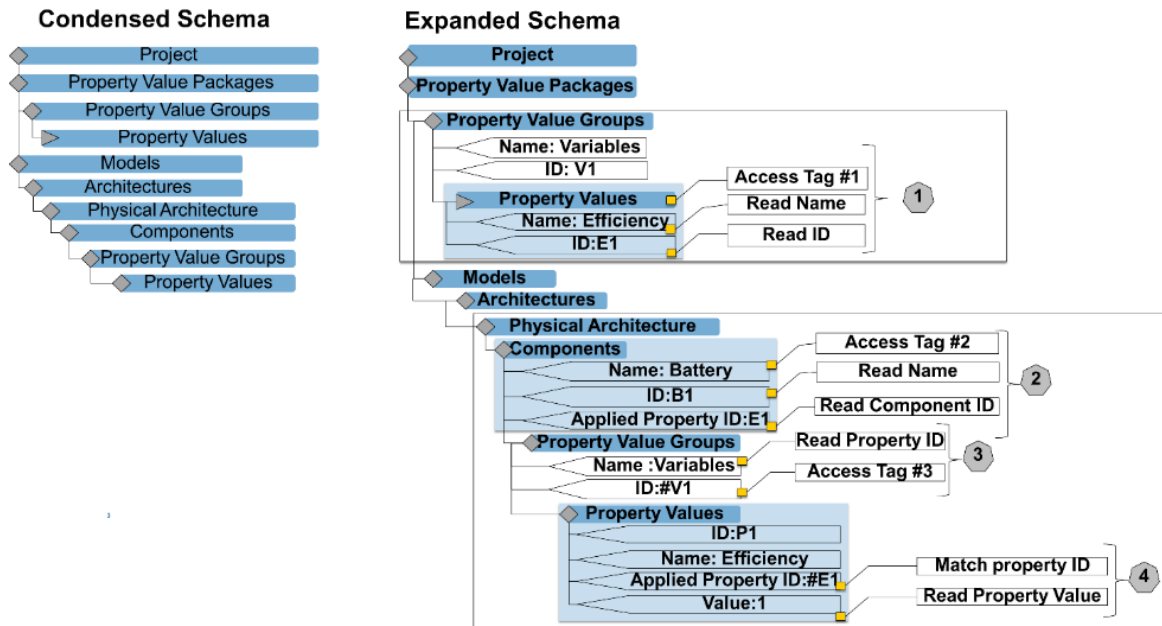


Figure A- 5: Condensed and expanded versions of the simplified Capella XML Schema

Figure A-5 shows the simplified Capella data schema with the tag names that represent different types of model information. An expanded version is also included with annotations describing the steps required to extract property information from the schema. These steps are labeled from 1 to 4 and include sub-steps about accessing and opening specific elements of each parent tag.

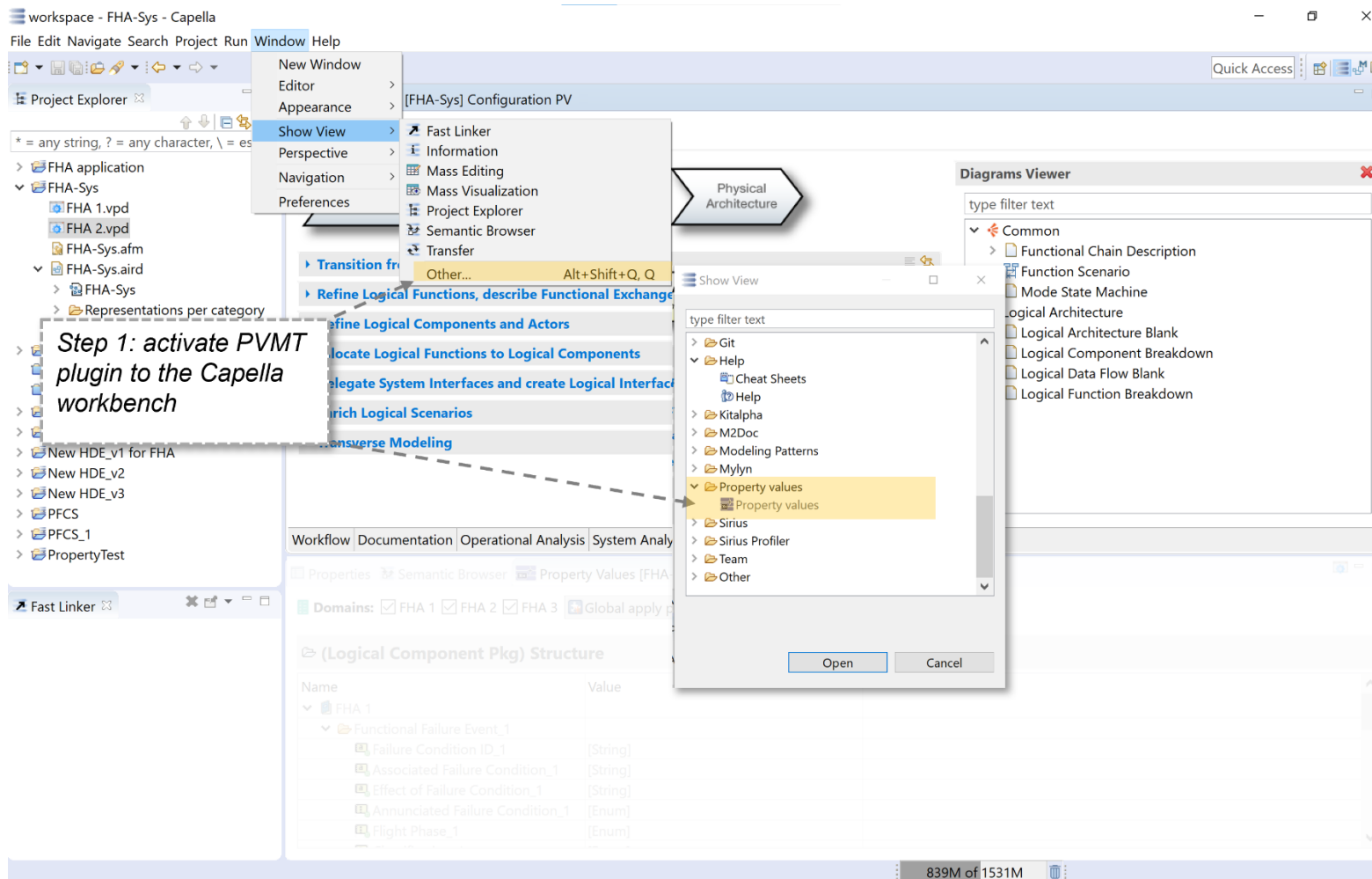


Figure A- 7: Activating PVMT add-on to a Capella project

This Section presents a guide to creating a viewpoint in Capella using the PVMT tool to enrich components with customized viewpoints. As PVMT add-on is not enabled within a project by default, the first step is to activate and add the Property Values to the project view.

Figure A-8 specifies Viewpoint Editor's location, where the viewpoint configuration can be defined, and users can start building a property domain for the Capella model.

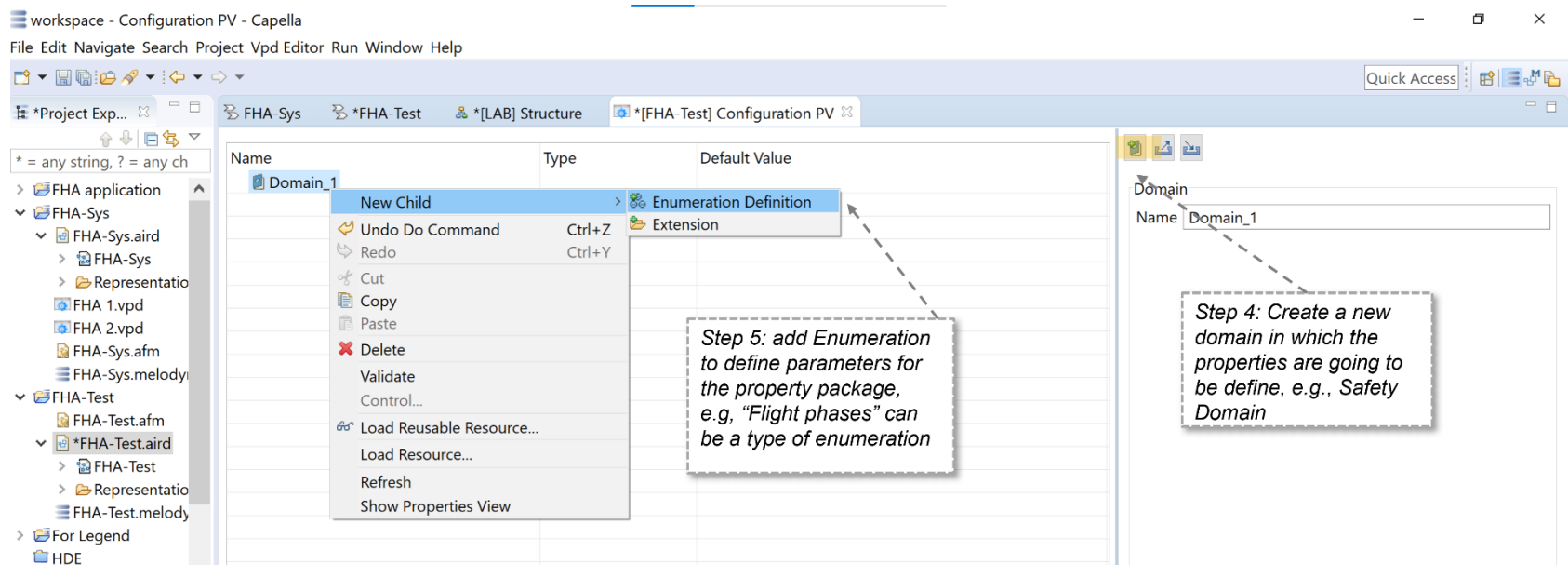


Figure A-9: Creating property domain in Viewpoint Editor

Figures A-9 and Figure A-10 show the steps of defining the property domain and adding initial parameters as enumeration. In the example of the safety property package, enumerations are defined as Flight phases. This feature can help users to specify the flight phase when a particular failure event occurs for a component or function.

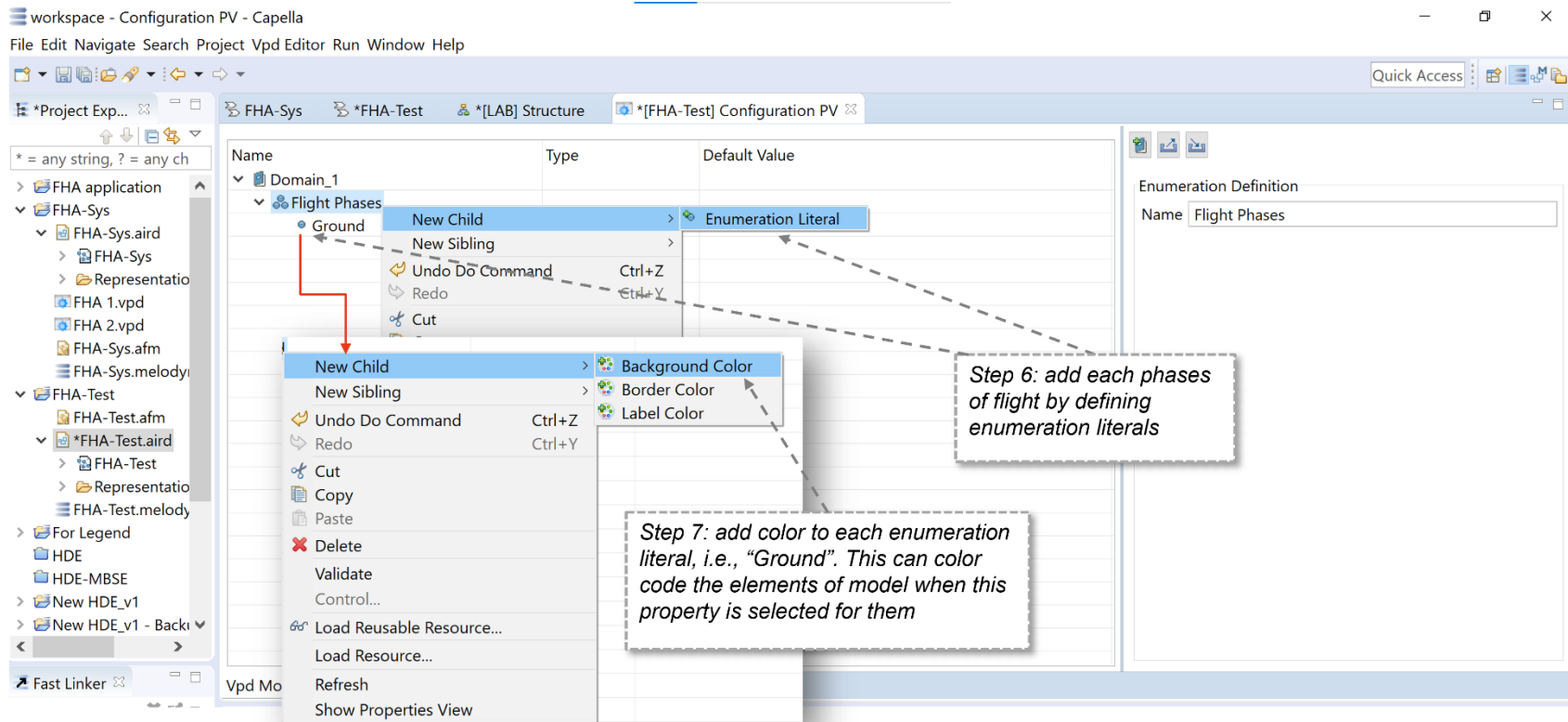


Figure A-10: Defining enumeration literal as parameters of properties

In Figure A-10, the enumeration literals are defined. The assignment of an enumeration parameter to a component can be made by selecting from the enumeration literal defined here. For example, each flight phase is defined as an enumeration literal.

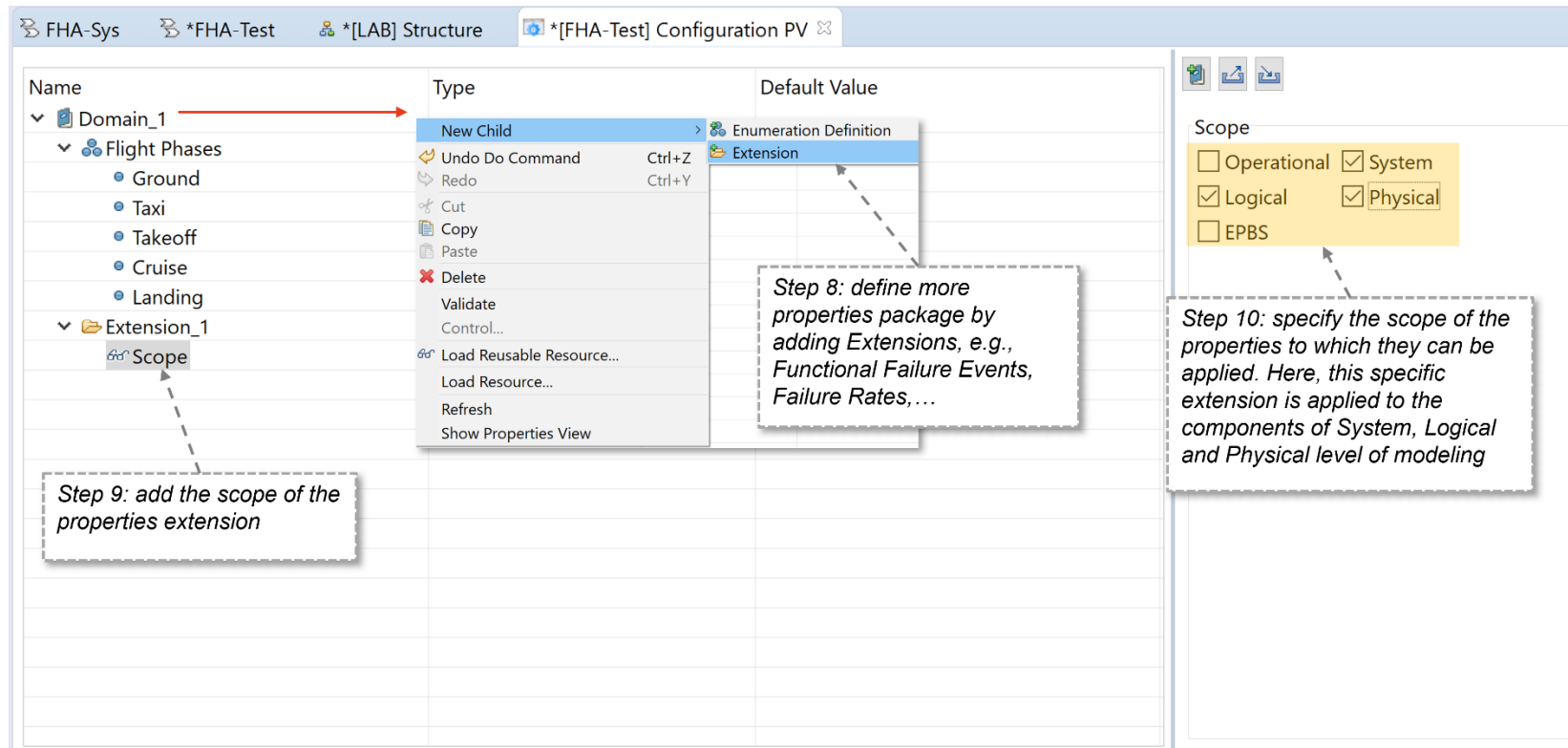


Figure A- 11: Defining the scope of the property domain and adding more parameters as properties

The next step is to define the scope of the property domain in which properties can be assigned. In Figure A-11, this property domain is to be applied for system, logical and physical components in Capella models. Adding extensions can define more properties, such as functional failure events and failure rates, for this property domain.

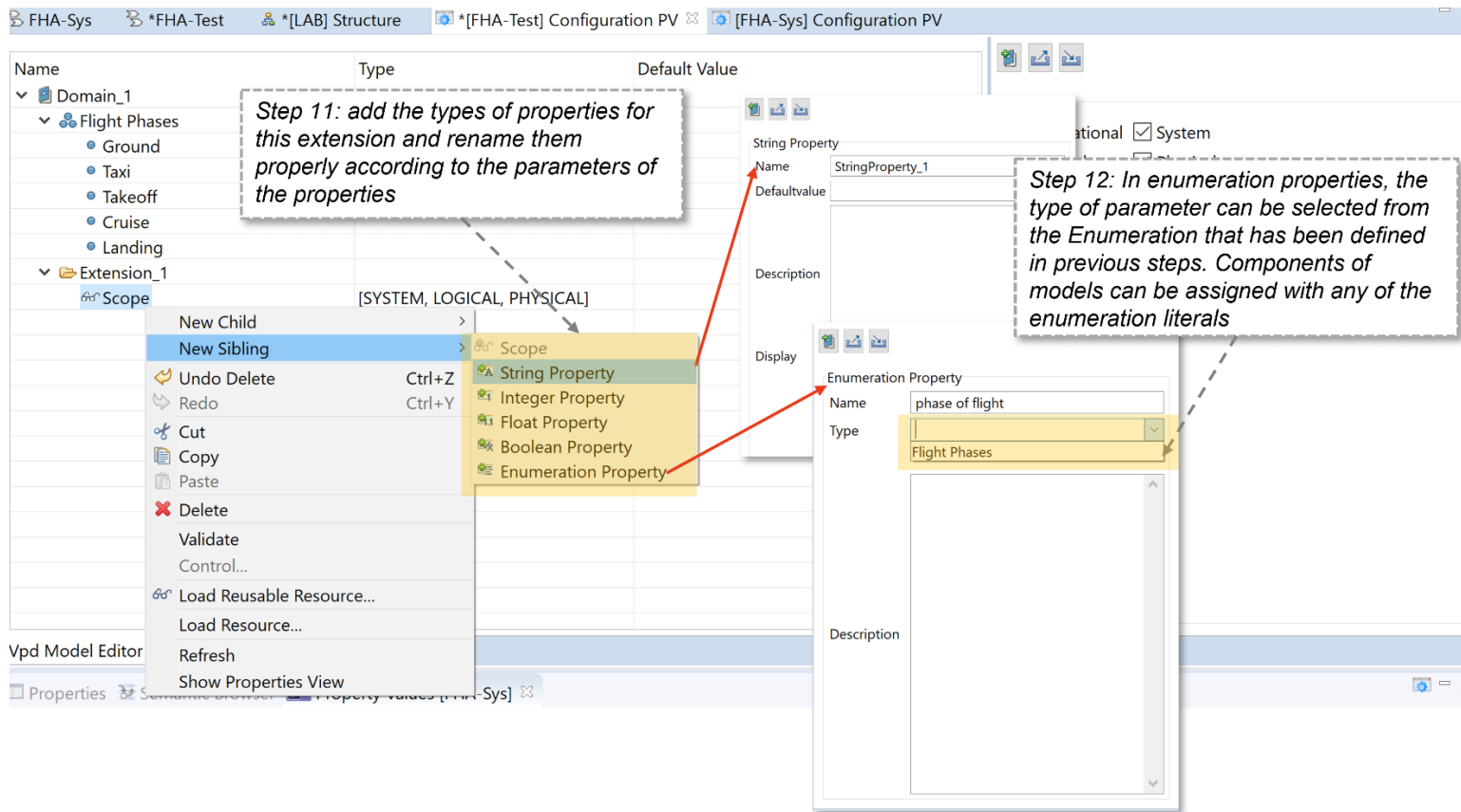


Figure A- 12: defining different property types for the extensions

This Figure illustrates how different types of properties such as String, Integer, Float, Boolean, and Enumeration. In each extension, any property of any type can be defined. Notably, for defining the enumeration property, users can choose from the enumerations defined in steps 5 and 6. Moreover, the property rule can be applied to specific properties. For example, users can specify the condition a property assigns to a component. These conditions can be “IF ” or value-comparing conditions. The complete safety property domain created to support the model-based FHA in the frame of this work is shown in Figure A-13.

Name	Type	Default Value
<div style="border: 1px dashed gray; padding: 2px; display: inline-block;"> <i>Enumeration Literal as classifications with assigned color</i> </div> Classification ←	Classification <ul style="list-style-type: none"> Catastrophic (Background Color: Red) Hazardous (Background Color: Orange) Major (Background Color: Yellow) Minor (Background Color: Light Green) No Safety Effect (Background Color: Blue) 	
<div style="border: 1px dashed gray; padding: 2px; display: inline-block;"> <i>Enumeration Literal as phases of flight</i> </div> Flight Phase Parameter ←	Flight Phases <ul style="list-style-type: none"> Stationary Take-off Initial Climb Climb Cruise Expanded Operation Descent Approach Go Around Landing Taxi Ground Flight All Rejected Take-off 	
<div style="border: 1px dashed gray; padding: 2px; display: inline-block;"> <i>Enumeration Literal as True, False, N/A</i> </div> Annunciated Parameter ←	Annunciated <ul style="list-style-type: none"> True False N/A Annunciated or Unannunciated 	
<div style="border: 1px dashed gray; padding: 2px; display: inline-block;"> <i>Extension including, String and Enumeration properties</i> </div>	Functional Failure Event_1 <ul style="list-style-type: none"> Scope: [SYSTEM, LOGICAL, PHYSICAL] Failure Condition ID_1: stringProperty Associated Failure Condition_1: stringProperty Effect of Failure Condition_1: stringProperty Annunciated Failure Condition_1: Annunciated (Default: True) Flight Phase_1: Flight Phases (Default: Stationary) Classification_1: Classification (Default: Catastrophic) Impacted Failure Condition(s): stringProperty 	
Functional Failure Events and Failure Rates ←	Failure Rate_1_Catastrophic <ul style="list-style-type: none"> Scope: [SYSTEM, LOGICAL, PHYSICAL] Property Rule: Classification_1 (Default: = Catastrophic) Failure Rate_1: floatProperty (Default: 1.0E-9 FH) Failure Rate_1_Hazardous <ul style="list-style-type: none"> Scope: [SYSTEM, LOGICAL, PHYSICAL] Property Rule: Classification_1 (Default: = Hazardous) Failure Rate: floatProperty (Default: 1.0E-7 FH) Failure Rate_1_Major <ul style="list-style-type: none"> Scope: [SYSTEM, LOGICAL, PHYSICAL] Property Rule: Classification_1 (Default: = Major) Failure Rate: floatProperty (Default: 1.0E-5 FH) Failure Rate_1_Minor <ul style="list-style-type: none"> Scope: [SYSTEM, LOGICAL, PHYSICAL] Property Rule: Classification_1 (Default: = Minor) Failure Rate: floatProperty (Default: 0.001 FH) 	
<div style="border: 1px dashed gray; padding: 2px; display: inline-block;"> <i>Extension for each classification, including Rule and Float properties</i> </div>		

Figure A- 13: Safety viewpoint created using PVMT plugin in Capella to support model-based FHA