Politecnico di Torino

Master's Degree in Aerospace Engineering

MSc Aerospace Engineering Thesis

Best Supply Chain, Manufacturing and Aircraft Systems identification leveraging a Model-based Product Line Engineering Approach

An integrated Product Line Engineering and Model Based System Engineering approach

Supervisors:
Dr. Roberta Fusaro
Dr. Davide Ferretto
Dr. Giuseppa Donelli (DLR)

Candidate:
Gabriele Allegra

A.y.2024/2025

Abstract

Public perception of aviation is a key driver of demand for enhanced environmental sustainability, with the aim of mitigating the industry's effects on climate change. The Flightpath 2050 initiative reflects this perception by aiming for environmental neutrality, global leadership, and societal responsiveness(Flightpath 2050, 2011). Achieving these objectives requires a shift in the approach used to the design of aeronautical systems, particularly concerning the integration of manufacturing and supply chain processes. The systems for system approach (S4S) aims to interconnect these systems from the conceptual design phase, enhancing integration and efficiency while minimizing costs and time losses. The objective of this thesis is to propose a methodology that leverages concepts derived from Model-Based Systems Engineering (MBSE) and Product Line Engineering (PLE) to generate all possible S4S architecture variants. As a consequence, all the possible combinations, the architecture, obtained by changing variables related to the aircraft design (e.g. number of components), manufacturing (e.g. materials), and supply chain are created.

In detail, the process starts with the identification of the stakeholder's needs, which are then translated into systems requirements. It continues with the development of a system architecture baseline. Variability is then introduced into the model by modifying materials, processes, and enterprises, driving the generation of the 150% model. Multiple variants of baseline architecture are then generated and compared in terms of aircraft, manufacturing, and supply chain performance (e.g. mass, production time). The best architecture is finally identified, based on the production and design criteria, by leveraging the value model theory. The approach proposed in this thesis enables a value-driven trade-off analysis that considers the performance of the aircraft system and the manufacturing and supply chain systems simultaneously, seeking the optimal solution for stakeholders and the overall performance of the three systems. An aeronautical case study demonstrates the benefits of applying the proposed methodology.

La percezione del settore avionico da parte dell'opinione pubblica è un fattore chiave per comprendere i motivi di un maggiore richiesta di sostenibilità ambientale, con l'obiettivo di mitigare gli effetti della aviazione sul cambiamento climatico. L'iniziativa Flightpath 2050 riflette questa percezione puntando alla neutralità ambientale, alla leadership globale e alla capacità di risposta della società (Flightpath 2050, 2011). Il raggiungimento di questi obiettivi richiede un cambiamento nell'approccio alla progettazione dei sistemi aeronautici, in particolare per quanto riguarda l'integrazione dei processi di produzione e della catena di fornitura. L'approccio "systems for system" (S4S) mira a interconnettere questi sistemi fin dalla fase di progettazione concettuale, migliorando l'integrazione e l'efficienza e minimizzando i costi e le perdite di tempo. L'obiettivo di questa tesi è proporre una metodologia che sfrutti i concetti derivati dalla Model-Based Systems Engineering (MBSE) e dalla Product Line Engineering (PLE) per generare tutte le possibili varianti di architettura S4S. Di conseguenza, vengono create tutte le possibili combinazioni dell'architettura ottenute cambiando le variabili relative alla progettazione del velivolo (ad esempio, il numero di componenti), alla produzione (ad esempio, i materiali) e alla catena di fornitura.

In dettaglio, il processo inizia con l'identificazione delle esigenze delle parti interessate, che vengono poi tradotte in requisiti di sistema. Prosegue con lo sviluppo di un'architettura di base del sistema. La variabilità viene quindi introdotta nel modello modificando i materiali, i processi e le imprese, portando alla generazione di un modello al 150%. Vengono quindi generate più varianti dell'architettura di base e confrontate in termini di prestazioni del velivolo, della produzione e della catena di fornitura (ad esempio, massa e tempi di produzione). L'architettura migliore viene infine identificata, sulla base dei criteri di produzione e progettazione, sfruttando la teoria dei modelli di valore. L'approccio proposto in questa tesi consente un'analisi di trade-off orientata al valore che considera le prestazioni del sistema aeronautico e dei sistemi di produzione e di supply chain simultaneamente, cercando la soluzione ottimale per gli stakeholder e le prestazioni complessive dei tre sistemi. Un caso di studio aeronautico dimostra i vantaggi dell'applicazione della metodologia proposta.

Contents

List of Figures

List of Tables

1 Introduction

The public perception of aviation has changed in recent years. Particular attention is now being paid to the quality, price, and speed of transport connections and to the impact aviation has on the environment and the noise levels emitted. The FlightPath 2050 links public priorities with the roadmaps of the various aviation stakeholders in a single vision to set milestones that will help transform and evolve today's aviation (Flightpath 2050, 2011). The goal? To achieve climate neutrality, prioritize customers' needs and be a leading example.

Research is a key process in achieving these goals, through which crucial technological breakthroughs can be researched and developed for cleaner, more sustainable aviation. The search for new technological breakthroughs alone is not enough to achieve the goals set. A review of how aviation systems are designed and built is necessary. One proposed change is the integration of manufacturing and supply chain systems within the conceptual phase of a product (Donelli Giuseppa, 2024). This change would make it possible to identify risks, unforeseen costs, and availability of materials and processes before obstacles are realized that could lead to delays, increased costs or lower product quality. At the same time proposing better collaboration with global suppliers, manufacturing companies, and regulatory agencies.

This approach makes it possible to design components and systems that are not only feasible but also less costly than a traditional approach (Donelli, Ciampa, et al., 2023a). Design knowledge of bottlenecks or slowdowns due to the non-feasibility of certain solutions makes it possible to build a system that bypasses the problems or addresses them upstream(Tang et al., 2009a). Thus, avoiding delays or redesigns during the production phase. Better synergy between stakeholders would reduce waste and increase efficiency throughout the life cycle of the system, and this would reverberate in future systems that could be lighter, less costly, and with higher performance. The result for the aviation sector is increased competitiveness, lower environmental impact, and technological advancement.

Considering manufacturing and supply chains at an early stage of the system aligns with the goals of Flightpath 2050. The integration of the three systems is necessary to overcome the limitations of the traditional sequential approach. In the traditional approach, the conceptualization of the manufacturing and supply chain systems has always taken place downstream of the aircraft conceptualization and when development begins. This approach has some downsides the main one is not considering manufacturing and supply chain a part of the knowledge that the designer could have immediate access to. The result is that any defects in the system are carried over to the next stages and the costs to resolve these defects increase as the system life cycle advances as shown in Figure 1.

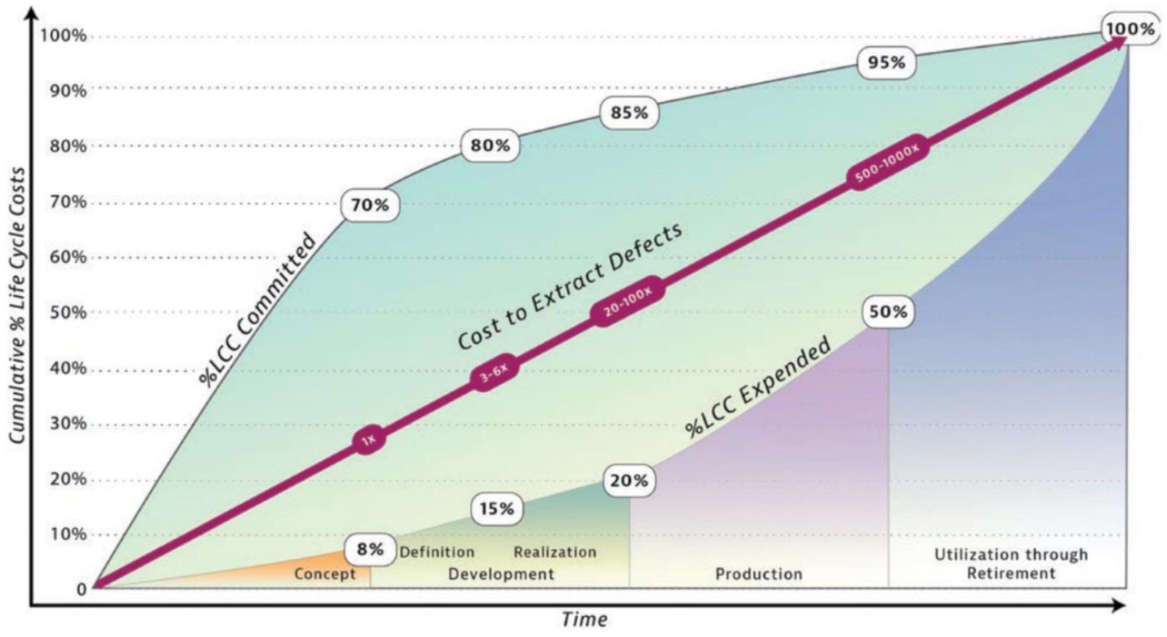


Figure 1: How costs to address defects increase in each step of the system life-cycle(INCOSE, 2023)

In the history of aviation, many examples of projects that suffered sudden stoppages due to the disregard of supply chain and manufacturing: In the case of the Boeing 787 Dreamliner which suffered production bottlenecks and delays due to design changes(Mocenco, 2015)(Tang et al., 2009b), or in the case of the Concorde by not taking into account the technological development of many parts, as the aircraft aged, became increasingly difficult to source until this factor contributed to its withdrawal from service(*What Was the Problem with Concordia?*, n.d.). Other industries have also experienced negative effects, as in the automotive sector(Canis, 2011).

Following the 2011 Fukushima tsunami, Toyota and Honda suffered production slowdowns and consequent loss of profits due to a lack of flexibility in the supply chain system in response to a localized event (Canis, 2011; Matsuo, 2015). By introducing supply chain and manufacturing system experts, possible limitations or constraints, such as those presented in the examples, can be identified earlier and thus enable the designer to avoid running into such problems.

The aim of this thesis is therefore to verify that the presented methodology results in time and cost savings leading to a better product.

The research is presented in chapters:

- Chapter 2. Literature review: it presents the current studies about this topic and systems engineering approaches that are used in Chapter 3. It also shows the gap in the research where this thesis is placed.
- Chapter 3. Methodology formulation: its objective is to explain step-by-step the entire methodology.
- Chapter 4. Methodology implementation: presents the tools and software used to implement the methodology presented in Chapter 3.
- Chapter 5 Methodology application: introduces the reader to a case study used to test methodology. Here, the results of the test case are offered.
- Chapter 6 Conclusions: in this chapter, a summary of the objectives achieved is reported together with some suggestions for future steps.

2 Literature review

This section reports the literature review currently available regarding the two approaches of systems engineering that are exploited in the following sections. Section 2.1 defines the Model-Based System Engineering approach, and its origins, history, and applications are reported. The same is done for the Product Line Engineering approach in Section 2.2.

2.1 MBSE approach

In recent decades, engineering projects in every field have grown in complexity and integration. Since this trend began, engineers have been facing the task of rethinking their approach to developing them (Ramos et al., 2012). One result of this challenge is MBSE (Model Based System Engineering). The term MBSE was first introduced by Wymore in 1993, but the initial idea differed from what we have today. The term included mathematical ideas such as algebraic relationships and mathematical rules for writing requirements (Madni & Sievers, 2018). Today, the approach is further away from the mathematical field MBSE is used to create models that link requirements, design components, analysis, verification, and validation tests (Tepper, 2010).

MBSE is defined as a system engineering approach with the purpose of creating and exploiting a model of a system that represents the primary source of information and the place where this information can be exchanged between designers, engineers, and stakeholders (Friedenthal, 2014). Using models to capture the fundamental element of a system enables better understanding, communication, and management of complex systems. This approach is particularly suited when paired with complex engineering projects, where the interactions between components and subsystems are particularly complex, and extreme coordination has to be implemented to achieve the desired outcome (Cloutier, 2010).

2.1.1 Document-based approach vs MBSE

The Document-Based Approach (DBA) for systems engineering is a traditional approach that relies on the use of formal documents, such as reports, to represent and communicate information about the system under analysis. Although DBA has been widely used it has several limitations, one of the main problems is update changes, in fact, changes to one document often require manual updates in all related documents. This task increases the risk of errors and inconsistencies (Walden et al., 2015). Moreover, documents are static tracking changes, and communicating them to other teams becomes complex, leading to increased costs, delays, and reduced quality of the final product (Friedenthal, 2014). The lack of an integrated view of the system can also make it difficult to identify and resolve issues before they become critical. A traditional approach flow is represented by the flow below, Figure 2

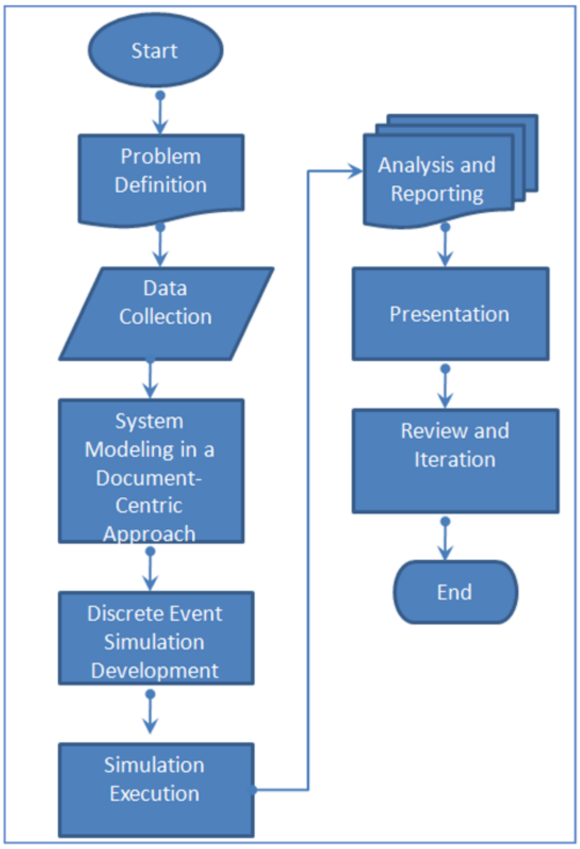


Figure 2: Document System engineering approach (Dutta Banik & Sengupta, 2024)

When a great mole of documents needs to be updated errors can be made that lead to costly rework, schedule delays, and late-stage changes. Errors made in capturing requirements, in system analysis and architecting phases, and system design, sometimes are only noticed when problems appear during the system acceptance phase, as shown in Figure 3.

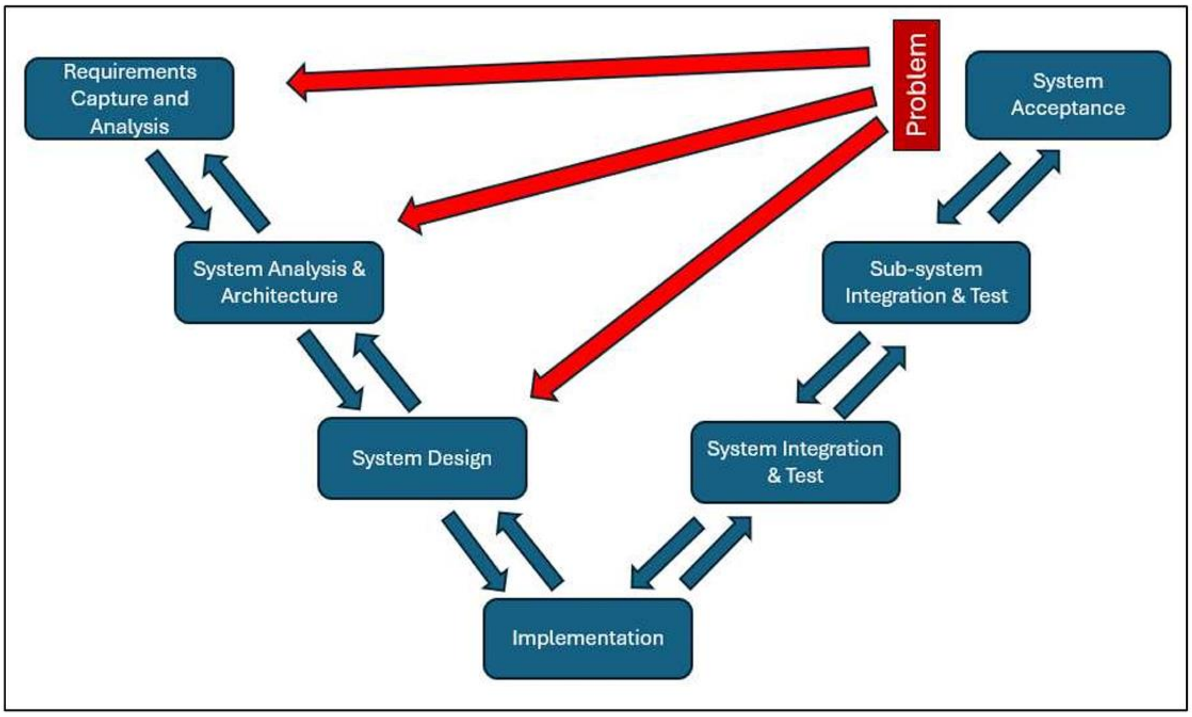

Requirements Capture and Analysis
Problem
System Acceptance
System Analysis & Architecture
Sub-system Integration & Test
System Design
System Integration & Test
Implementation

Figure 3: Problem detection linked to errors made in previous steps (Siddique, 2025).

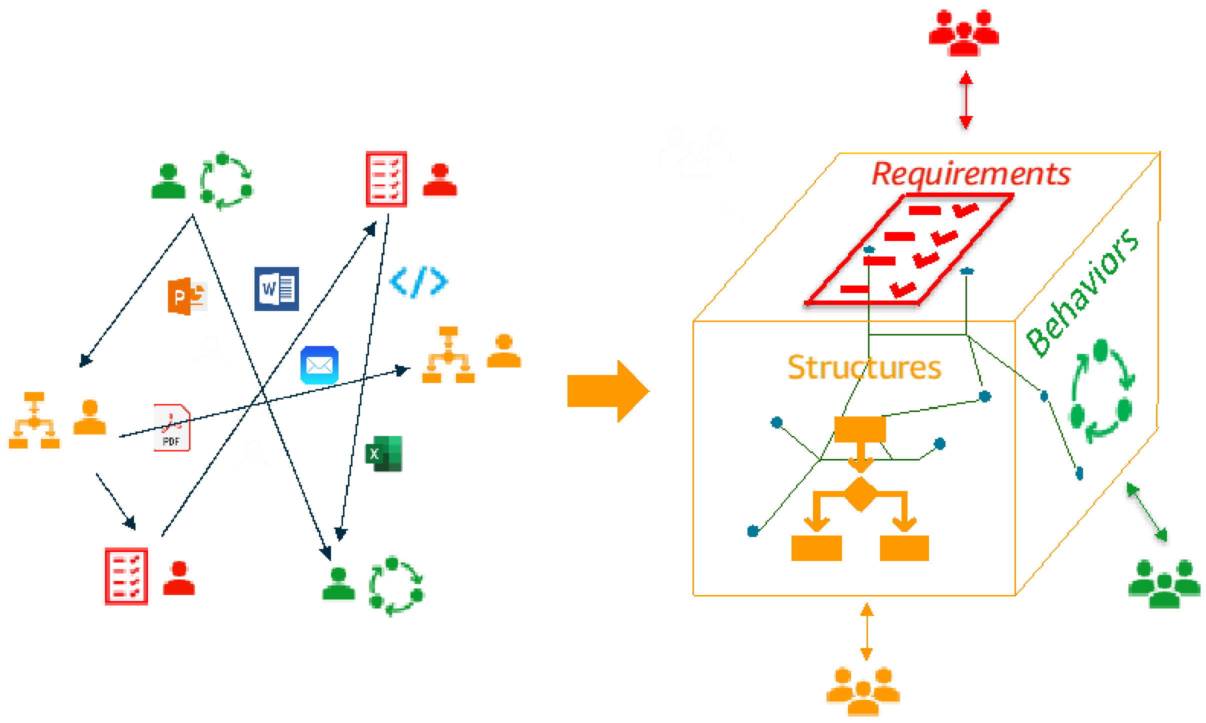


Figure 4: Differences between Document based engineering and MBSE (AWS, n.d.)

The transition from a document-based approach, in which system specifications and designs are dispersed across multiple documents, to a model-based approach that integrates a shared system model helps in enhancing collaboration and mitigating the errors made. This approach is particularly useful for complex systems. Although MBSE offers many advantages, its implementation requires specialized skills and tends to increase the initial complexity (Friedenthal, 2014).

2.1.2 Motivation to use MBSE

In MBSE, a model is a 'single source of truth' meaning that all important information is represented within the model (Madni & Sievers, 2018). Having all the information gathered in a single place is an advantage. It makes it easier to connect information, as well as the components of the system, and the result is that it is easier to find information quickly and find errors and inconsistencies. Moreover, since the model is the core of all information by update it changes are shared between all designers.

The interest in MBSE comes from the need to overcome known difficulties and undesirable practices. The best time to explore different solutions is at the beginning when the problem is first defined, but one of the major issues in systems engineering is building the wrong features at the beginning of the conceptual phase (London, 2011), a mistake like this can lead to a waste of time and money. Another problem is that engineers often start modeling without fully understanding if they are solving the right problem (R. Karban T. Weilkiens R. Hauber M. Zamparelli R. Diekmann A.M. Hein, 2011). If system architecture is not carefully planned problems can appear in later phases, such as when integrating different parts of the system. It is important to understand that as designers make decisions, the available options decrease and the cost related to decisions taken are locked even if the system is not developed. Designers often rush through this phase, generating costs and wasting time on later processes.

MBSE was introduced also to better manage requirements. Studies show that engineers spend a lot of time searching for information and creating reports, as systems become more complex, requirements increase, and the more they increase, the harder it is to manage them with simple tools like checklists or separate databases. So, traditional document-based approaches make it easy to miss important details and connections between system parts (J.S. Topper N.C. Horner, 2013). Because of this, many engineering teams are moving from document-based to model-based methods. MBSE allows engineers to generate up-to-date system documents automatically.

One of the biggest challenges in system engineering is communication. Many different specialists, like designers, users, testers, and stakeholders need to share a common understanding of the system. This requires clear documentation, clear requirements and needs, and documentation about system interfaces. This shared information helps creating a common understanding of the system to improve collaboration, and MBSE helps achieve it. Traditionally, system information is stored in a multitude of different documents that need to be updated every time a change is made. However, when working with many stakeholders or complex systems, these documents can easily become inconsistent or outdated. MBSE solves this problem because the system is a model, and every update is shared with other designers and stakeholders, this way the information is always up-to-date (Madni & Sievers, 2018).

A strategy utilized starts with simple components and gradually builds more complex subsystems until eventually the full system is developed. The model can store information, such as relationships between subsystems, system behavior, and requirements, in an organized way. Since all information is linked it is easy for engineers to track decisions, analyze changes, and maintain consistency.

2.1.3 State of the Art

Estefan's survey can be read to understand the latest developments in MBSE (Estefan, 2007), it describes different MBSE methods and introduces Systems Modeling Language (SysML) (Weilkiens T. et al., 2016). MBSE has been widely used in system architecture development. During this phase, system's structure, behavior, and main functions are derived. MBSE helps to visualize these system elements easily and also it helps to connect them to stakeholders' requirements. This is the reason why both in the USA and in Europe, MBSE is starting to be more used.

More experts every year believe that MBSE is a powerful way to improve speed, reduce costs, and increase quality (Paredis Christiaan, 2011). The use of MBSE has shown that it can shorten development time and reduce errors, this is partly because it helps engineers better understand the problem from the beginning, as stated before this is a common issue. Additionally, studies suggest that traditional document-based methods capture only about 50% of the problems (Jorgensen, 2011), instead using MBSE techniques, like use cases, can increase understanding to over 90%. But still little data is available to demonstrate what is said, and even less data is available to check the actual time savings. However, studies are carried out to prove MBSE's Value (Estefan Jeffrey, 2011).

MBSE is growing quickly. INCOSE (the International Council on Systems Engineering) has an active MBSE working group that continues to grow. In 2013, the IEEE Systems, Man, and Cybernetics Society created the MBSE Technical Committee intending to share research findings. This committee is working closely with INCOSE.

The standardized language SysML is being studied to be improved and to identify any gaps in how models represent systems. MBSE is also becoming part of professional and academic education, major aerospace companies like Lockheed Martin, Boeing, and Northrop Grumman now offer MBSE courses. From the academic point of view programs at leading universities, including MIT, Georgia Tech, USC, Johns Hopkins, and the University of Tokyo, have added MBSE to their curricula. MBSE is now so important that it is recognized as a best practice in systems engineering at major research institutions and federally funded labs, such as NASA's Jet Propulsion Laboratory, Sandia National Laboratories, NASA Glenn Research Center, and MITRE Corporation (Madni & Sievers, 2018).

2.1.4 System model characteristics

In MBSE, the model is initially only a basis for the final system. As work on the model continues, the abstract model becomes more and more concrete and what was initially only an abstraction of reality, perhaps starting only from an analysis of the system's behavior, eventually becomes a model that represents reality in which the hardware and software components are also modeled.

Modeling begins with the key questions that the model must reply to. These questions drive system initial behavior, but it may change in the course of system development. A model is considered “fit for purpose” when it can reliably answer all the relevant questions posed above. As the model changes its behavior, components, or the way it answers the questions asked over time, it is considered a 'living representation' of the system (Madni & Sievers, 2018).

A model can describe a system at different levels of detail or abstraction. Indeed, it can be simplified by ignoring certain details that are not relevant to its purpose, this flexibility allows different stakeholders and designers to visualize and focus on what part of the system is important to them. For example, a structural engineer may care about the size and weight of electronic components but not their function, while a software engineer focuses on functionality instead of physical details.

Models can take different forms, and some of them are informal, like sketches or basic concept diagrams, while others are formal and use precise and standardized representations such as SysML diagrams or state charts (Friedenthal, 2014). Formal models can be purely descriptive (static) or executable (dynamic), meaning they can simulate system behavior.

If used correctly models can improve understanding, communication and at the end stakeholder decision-making. They help visualize complex systems, track requirements, and support verification and validation. However, poorly used models can lead to incorrect conclusions. As statistician George Box (Box George E.P., 1987) said, “All models are wrong; the practical question is how wrong they have to be to not be useful.”

Two important concepts in modeling are verification and validation.

- Model verification checks the model to see if it is correctly built. To be a verified model it should be complete (cover all necessary details), consistent (use the same definitions and assumptions), and traceable (linked to requirements and standards).
- Model validation ensures that the model accurately represents real-world behavior and follows relevant theories, data, and regulations. A model is tested to see if it behaves as expected. If it fails, it is adjusted and tested again in an iterative process that builds confidence in its accuracy (Madni & Sievers, 2018).

2.1.5 Future Trends

The system engineering field is currently evolving, and MBSE is evolving as well. Some new trends are emerging to update MBSE capabilities, one of the most significant is MBSE integration to AI (Artificial intelligence), the Internet of Things (IoT), and the idea to develop digital twins (Heydari S.A., 2023).

The contribution of artificial intelligence to system models can guarantee a greater capacity for analysis and decision-making of the system and should also be able to automate the generation of certain models, optimize them, and test the behavior of these systems under adverse conditions that could not be tested before. All these new capabilities should allow for a reduction in the time for system creation, thereby lowering costs (Gore, 2020). The integration of MBSE with IoT, on the other hand, is particularly interesting for those industries where the ability to update systems and monitor them is particularly important, such as the aerospace industry (Holt, 2010). This integration would allow a large amount of data to be collected from systems and analyzed them, so that the system can be updated faster than could have been done before.

Finally, the representation of real systems, or digital twins, combined with the capabilities of the MBSE, would allow more accurate and detailed models to be created, especially for complex systems. The impact would be significant, especially in the testing part of the model, and could be closer to the final system and thus reduce errors and risks (Gore, 2020).

All these new technologies, when combined with the already great capabilities of MBSE would reduce errors, time, and risks, making MBSE an indispensable tool (Heydari S.A., 2023).

2.1.6 Missing capabilities

Despite MBSE's great capabilities in clarifying and simplifying the understanding of the system and connecting its elements. MBSE is only capable of creating one model of the system. If more than one instance is created, several models must be implemented in parallel. The rapid creation of multiple instances is a necessity, which is why extensions to the SysML language (Bussemaker et al., 2024), the basis of MBSE, has been implemented to add variability to the system. Examples are CVL (Broodney et al., 2012) and VAMOS (Weilkiens, 2015), or more recently SysML v2 in which variability is a key point (Bajaj et al., 2022) demonstrating how important it is for future developments. Other attempts include including variability using feature models (Gedell & Johannesson, 2013).

2.2 PLE approach

Product Line Engineering (PLE) is a methodology used in system and software engineering. Its main objective is to efficiently manage a family of systems based on shared elements. At its core PLE enables the creation of different systems by sharing and managing assets and then combining them to generate a system based on market requests o specific needs from clients. By leveraging this approach time can be saved and limited costs are needed to develop costumed solutions (INCOSE, 2021). In the last decades, this approach has been more used thanks to the market need for these costumed systems.

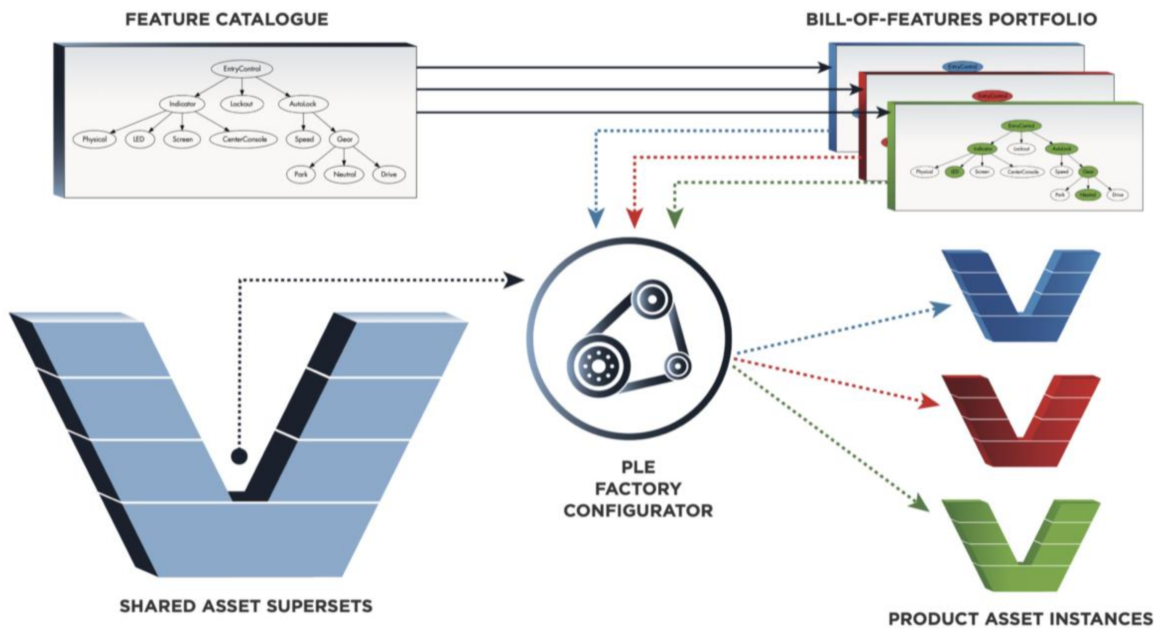


Figure 5: PLE approach in a scheme (Engineering, 2022)

PLE defines a shared set of assets that can be systematically reused across multiple systems. The Feature Catalogue is a structured list of all possible features that can be included in a product line. Each product within the product line is defined by a Bill of Features (BoF), which specifies the exact combination of features selected from the Feature Catalogue. A Shared Asset Superset includes all reusable assets, these shared assets are configured based on the features selected in the BoF. The connection between these elements is depicted in Figure 5

2.2.1 PLE origins and development

The PLE approach has its origin in the manufacturing industry. The need was to reduce costs related to the development of a system that shared common components. Then this approach has been extended to software solutions. In the last decade of the last century software industry started to formalize the “software product lines” concept, the objective once more was to reduce costs and improve the reusability of software development. Then this approach evolved until it touched software-hardware systems that were complex like the ones that can be found in the automotive and aerospace industries(Clements & Northrop, 2001). Once this approach became more interesting and used, new tools were introduced to better design and manage product instances. These tools enable the companies to have an instrument to develop product variability systemically. At the integration with MBSE was natural to have a better methodology to manage variants.

2.2.2 State of the Art

Nowadays PLE approach is used in the aerospace industry where systems are complex, and it is important to manage them. Airbus and Boing are two examples of companies that leverage the PLE approach to develop families of products (aircraft) that all share some common components or common structures(INCOSE, 2022). Using this approach to generate these aircraft allows us to save development costs and even maintenance costs. The A320 Programme by Airbus is a state-of-the-art example. This project includes different variants (A318, A319, A320, A321) that all share a common base(Forlingieri, 2022; Forlingieri & Weilkiens, 2022a, 2022b). On the other side of the ocean, Boeing has also adopted similar strategies with its 737 and 787 projects. The results are different aircraft configurations that share the same fuselage module or similar avionic systems (Wiley, 2023).

In the military field, another virtuous example is the Lockheed-Martin F-35, the different variations of this fighter share common structural and software components. This results in an impact on maintenance for USA military forces and allies (Springer, 2023).

2.2.3 PLE and MBSE connection

PLE and Model-Based Systems Engineering (MBSE) are two complementary approaches that improve the management of the complexity of modern engineering systems. As stated before, PLE focuses on the management of product variants and the reuse of components as much as possible. Meanwhile, MBSE approach uses system digital models to support the design, development and verification of the product under development.

The last INCOSE work highlights how the integration between PLE and MBSE allows the improvement requirement's traceability and management configuration of variants in a more efficient way (Madeira et al., 2023). Using modelling tools such as SysML, it is possible to graphically represent the variants of the system and analyze them to understand the impact of changes in system life cycle.

A significant case study is the Boeing 787 Dreamliner project where PLE and MBSE were used. For this project, Boeing used digital models to optimize the design of the different variants of the aircraft, by doing so it was able to reduce development costs and improve system reliability. This synergy has allowed Boeing to maintain a high level of quality and safety for its product (Wiley, 2023).

3 Methodology Formulation

This thesis aims to present a methodology to find out the best solution for a Systems for system, featuring as a System of Interest the aircraft and as Enabling Systems the manufacturing and the Supply chain systems, leveraging MBSE and PLE. Model-Based System Engineering is chosen as the methodology to formulate the system structure, while the Product Line Engineering approach is used to generate a family of system variants in preparation for the MAUT analysis. To select the best solution for the S4S, this study considers both the performance of the aircraft and the production line (supply chain system and manufacturing system combined). MAUT analysis enables the evaluation of multiple solutions for the same system thereby enhancing knowledge of the system before committing capital through a specific decision that will influence the development process and further steps.

The methodology starts with the collection of stakeholder needs, identifying key areas of interest that must be translated into requirements by the designer (Boggero et al., 2020). Subsequently, once the stakeholders' and systems' requirements are gathered, the process continues with the modeling of the system architecture for the three systems under analysis. The PLE approach is then applied to this baseline design resulting in the generation of a family of architectures for the same system.

Finally, these architectures are evaluated using a value-driven approach and a trade-off analysis is performed to select the optimal solution. This evaluation considers the performance of the system of interest (the aircraft), the performance of the two enabling systems, and the stakeholders' interests(Donelli, Boggero, et al., 2023a).

The figure below schematically illustrates the proposed methodology.

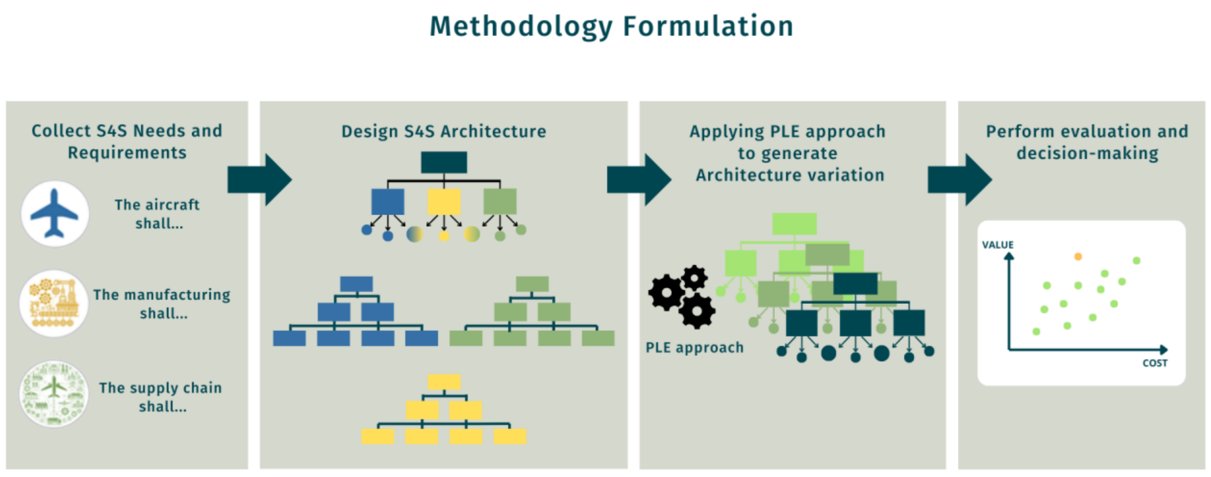

Methodology Formulation
Collect S4S Needs and Requirements
The aircraft shall...
The manufacturing shall...
The supply chain shall...
Design S4S Architecture
Applying PLE approach to generate Architecture variation
PLE approach
Perform evaluation and decision-making
VALUE
COST

Figure 6: S4S methodology formulation leveraging MBSE and PLE approaches

The methodology is linked to a test case that is presented in Section 4. A small preview is reported to simply the understanding of the theory. In the case under analysis, the system of interest is Horizontal Tail plane of an aircraft, it is composed of four kinds of structural elements: ribs, stringers, skin panel and spars. The two enabling systems are the supply chain, composed of Suppliers Tier 1, Tier 2 and OEM facilities, and the manufacturing system that is composed of the machine that will perform the processes to build the HTP. The variability of the system is introduced considering different materials, processes and facilities.

3.1 Glossary

Before proceeding and delving into the core of the methodology it is necessary to introduce some key definitions and concepts that will recur throughout this thesis.

According to ISO/IEC/IEEE standards, a **system** is defined as: “An arrangement of parts or elements that collectively demonstrate properties, behavior, or meaning that the individual components do not.”(E. Freund, 2005)

A **System of Interest (SoI)** is defined as the system under analysis(E. Freund, 2005), development, and design. In this case, the SoI is the **aircraft**. The aircraft is described as a system whose purpose is to fly, to achieve this goal many components (e.g. wings, fuselage, landing gear, tail, etc.) are needed, each with distinct purposes. For instance, the horizontal tail plane is designed to control the longitudinal dynamic. The collective function of these components is to fulfill the primary function of the aircraft system.

The two systems, **supply chain,** and **manufacturing**, are classified as **Enabling Systems (ES)**(E. Freund, 2005), their role is to support and facilitate the SoI during the various phases of its life cycle, particularly during production(Donelli et al., n.d.). They are defined as follows: the supply chain system is a network or, cluster of companies and facilities, responsible for producing and assembling the components of the aircraft. The supply chain has the capability to produce the entire aircraft, meanwhile, a single facility may be capable of completing one or more components. Meanwhile, the manufacturing system is defined as the system consisting of machinery and processes that transform raw materials into finished components or subcomponents. The two systems are part of the aircraft's production. (Donelli Giuseppa, 2024)

The language usually used to design the model of a MBSE system is **SysML**. This language is used in this thesis and therefore some key elements have to be introduced. These definitions are taken from (A Practical Guide to SysML The Systems Modeling Language, n.d.; Weilkiens, n.d.):

- **Block** is an element that describes parts of the system's structure. It can represent a logical or a physical unit of the system. It is represented as a rectangle, it always has a name that determines the element described. It can be further characterized by **values**, **operations**, **constraints,** and **references**. A block can always be divided into other blocks, they are, also, parts of the main block.
- **Part** is an element of the system that is a child of a block. While blocks can only show the hierarchy between the units of the system. Parts are used to show the connection between the elements inside a block.
- **Ports** represent input and output of a part, and they represent the way the unit interacts with the world. Ports can be parts themselves or just a way to show how the part interacts with the connector.
- **Item Flow** is an item that flows between two parts, it is bond to the connector that connects the two or more parts considered. It can be an abstract element, like an event, or a physical object.
- **Actor** is a system, group of people, or others that interacts with the system, but it's not part of it.

- **Use Case** is a way to describe an interaction between the system and an actor. It describes a service or a function that the system performs. Usually, the actor starts the service via a trigger.
- **Activity** is a way to describe the actions that the system has to be able to do to perform a function or to arrive at a precise state.
- **Action** is an elementary step performed by the system or a system's component.

The definitions introduced form a basis to help the reader in the understanding of the following chapters. Obviously, these definitions are only some of the many presents in the SysML language, that's the reason why further are introduced when necessary.

3.2 S4S concurrent approach

The concurrent approach is designed to address the need to overcome certain limitations of the traditional sequential approach. Considering a typical life cycle, such as the one proposed in ISO/IEC/IEEE 15288:2023(E. Freund, 2005) and adopted here, a system progresses through six stages during its life span: conceptualization, development, production, utilization, support, and retirement. Some of these stages may overlap, occur in parallel or they may be absent depending on the system's functionalities. In the case under consideration, each system follows the six stages presented. The first stage, conceptualization, begins with exploratory research by the designer into the stakeholders' needs to identify both stakeholder requirements and technical system requirements. At this phase, these requirements are transformed into functions that the system must fulfill through its architecture. Then, multiple architectures are conceived and analyzed finally, various studies are conducted to justify the selection of the final configuration. Subsequently, the selected system architecture is developed while the requirements are refined until the prototype of the concept is validated. Then the system is implemented, utilized, maintained, and ultimately retired from use, marking the end of the system's lifecycle.

In the aerospace industry, it is standard practice for supply chain and manufacturing systems to be designed and developed only after the final definition of the system of interest has been determined and the development stage is started, as the red arrow illustrates in the image below(Donelli Giuseppa, 2024).

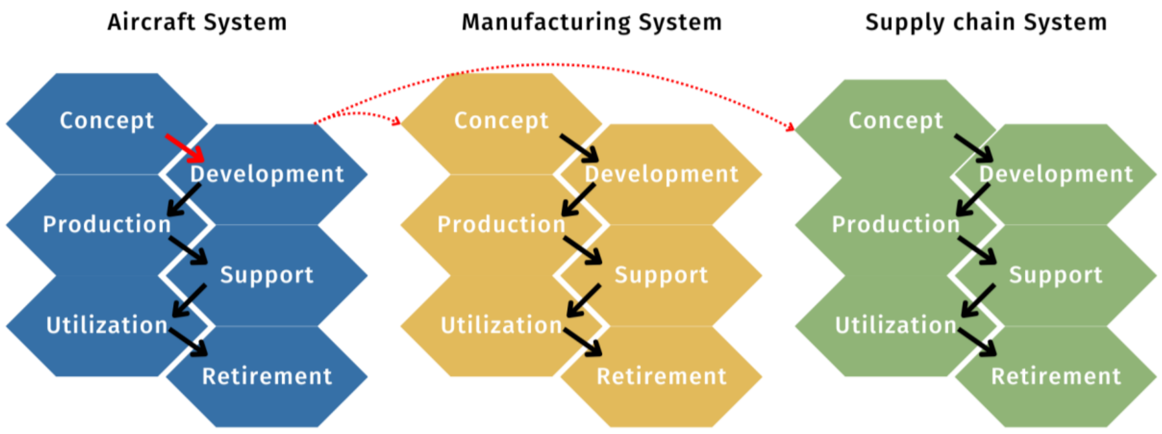


Figure 7:Traditional approach in the development of an aircraft system.

This approach allows the optimization of aircraft performance postponing analyses related to logistics, production cost evaluations, feasibility assessments for certain components, potential production issues, and supplier selection(Donelli, Boggero, et al., 2023a). All these limitations can lead to cost overruns, production delays, and the need for component redesigns.

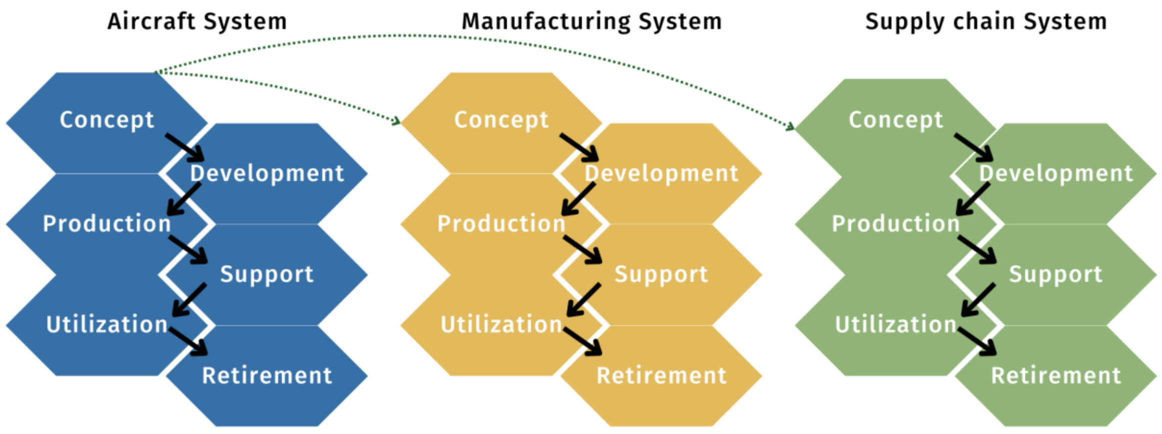


Figure 8: Concurrent approach in the development of an aircraft system.

The presented concurrent approach involves the conceptual design of the three systems being carried out in parallel, interactions are design aim to minimize or eliminate the disadvantages of the traditional approach. This approach enables the S4S to derive a solution that considers the strong points and limitations of all three systems.

During the conceptual stage, the traditional standard foresees 14 processes throughout the system lifecycle(E. Freund, 2005), but only 6 are related to the phase under examination (conceptual). The proposed S4S framework includes only 5 and these are applied in parallel to all three systems under consideration.(Donelli Giuseppa, 2024)

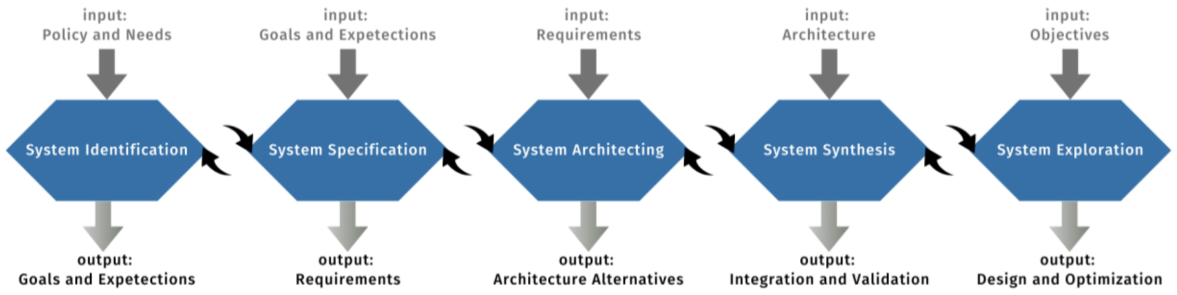


Figure 9: Processes included in the S4S framework

- System identification: This first phase of the S4S framework gathers the interests, needs, and expectations of the stakeholders regarding the system. To identify needs it is first necessary to identify all the system's stakeholders. A stakeholder represents an individual, a group of people, or an organization that has an interest in the system. For example, in the case of a passenger aircraft, the pilot, the airline, ground personnel, and regulations are all stakeholders of the system. Through interviews, stakeholders express their interests which are then gathered, but their needs, by definition, lack structure, rules, or patterns so they can be misunderstood and often they are confusing. For this reason, needs must be translated into requirements which, in contrast, have a clear structure, precise rules, patterns, types, attributes, and identifiers that allow the needs to be made explicit in a clear and unambiguous way, as well as providing limitations and guidelines for the system to be designed.
- System Specifications: As previously introduced, needs are translated into requirement expressions. Requirement expressions consist of the requirement statement and

attributes, for the former, it is expected that certain patterns and rules are followed; although each agency may choose different patterns to follow, it is important to maintain consistency in the choice made. In this work, the rules and patterns introduced by the INCOSE (International Council on Systems Engineering) guide are followed.

Requirement expressions are characterized by a type, using these types, it is possible to quickly identify the category of the requirement expression, additionally, for each type of the requirement pattern to follow also changes.

Finally, attributes are concepts that characterize a requirement expression. The main ones include ID, Parent Source, Means of Compliance, Attribute Type, Version, Owner, etc. At the end of this phase, a considerable number of stakeholder requirements and system requirements are derived, and they will drive the system design.

- <u>System Architecture:</u> Leveraging the previously collected requirements is possible to generate functions that the system must fulfil. The generation of functions using requirements explains how the requirements drive the design of the system. These functions have to be implemented by one subsystem or the integration of more subsystems. The result is a series of system architectures that meet the stakeholders' requirements and the derived requirements.
- <u>System synthesis and exploration:</u> Evaluation is undertaken based on indicators deemed significant by stakeholders, such as cost, risk, quality, and time. A trade-off between performance and costs is necessary during the decision-making phase, the objective is to select a final architecture, based on the stakeholders' needs. The chosen architecture can then proceed to the validation processes and only after it can enter the production phase. In the specific instance of the concurrent approach, the performance to be evaluated encompasses not only that of the aircraft itself but, also, that of the production system in its entirety, the supply chain system and the manufacturing process.

The present work focuses on the acquisition of the stakeholders' needs and requirements, and the architectural design phase, in particular, at the end of the architecture phase it proposes a new methodology for the formulation of multiple architecture variants, leveraging a combination of MBSE and PLE approach. Subsequently, the methodology is applied to a test case in order to verify the quality of the results. Finally, a value-driven analysis is applied to evaluate the different architectures and validate stakeholder requirements.

3.3 Needs and Requirements Formulation

This chapter deals with system identification and specification, and before the identification of requirements can begin, it is necessary to identify the relevant stakeholders.

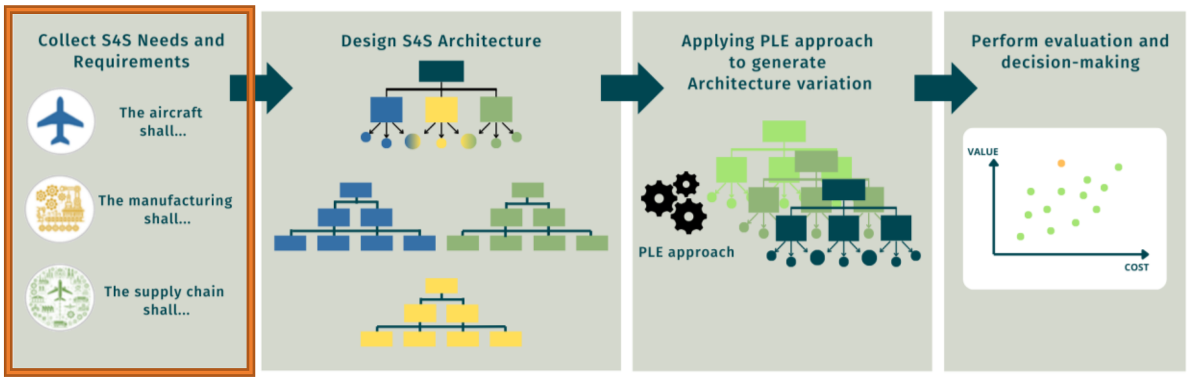

Collect S4S Needs and Requirements
The aircraft shall...
The manufacturing shall...
The supply chain shall...
Design S4S Architecture
Applying PLE approach to generate Architecture variation
PLE approach
Perform evaluation and decision-making
VALUE
COST

Figure 10: First methodology step

According to ISO 29148, "stakeholders are defined as individuals or organizations with a right, share, claim, or interest in a system or in its possession of characteristics that meet their needs and expectations"(Engineering Standards Committee of the IEEE Computer Society, 2011). The stakeholders to be identified span across the three systems and can range from government entities who prescribe regulations regarding groups of people working within or in contact with the system, to users of the system, to companies that create a product or a service for the system. Subsequently, to the identification of the relevant stakeholders, expectations regarding the three systems are collected through interviews (Boggero et al., 2020). This process focuses on the collection of information regarding the functions that stakeholders want the system to perform, the system's objectives, and limitations. Stakeholders' needs are essentially statements made in the form of requests, and for this reason, they lack structure and may be subject to misinterpretation(Ryan & Wheatcraft, 2022). This is the reason why they are then translated into requirements. Each need must be assigned an identifier and linked to the relevant stakeholder. This approach ensures that every need can be traced back to its respective stakeholders during the requirements writing phase. A requirement is defined as "a statement that translates or expresses a need and its associated constraints and conditions"(Engineering Standards Committee of the IEEE Computer Society, 2011), written in a language that can take the form of a natural language. If expressed in the form of a natural language, the statement should comprise a subject, a verb, and a compliment." The process of translating needs into structured expressions is referred to as "requirements", though they are technically termed "requirement expressions". Each requirement expression comprises a requirement statement and several attributes(Ryan & Wheatcraft, 2022). The former represents the text of the requirement, while the latter includes additional characteristics that ensure the traceability of the requirement to the need from which it originates, specify how it will be verified, identify the author, or provide a unique identifier.

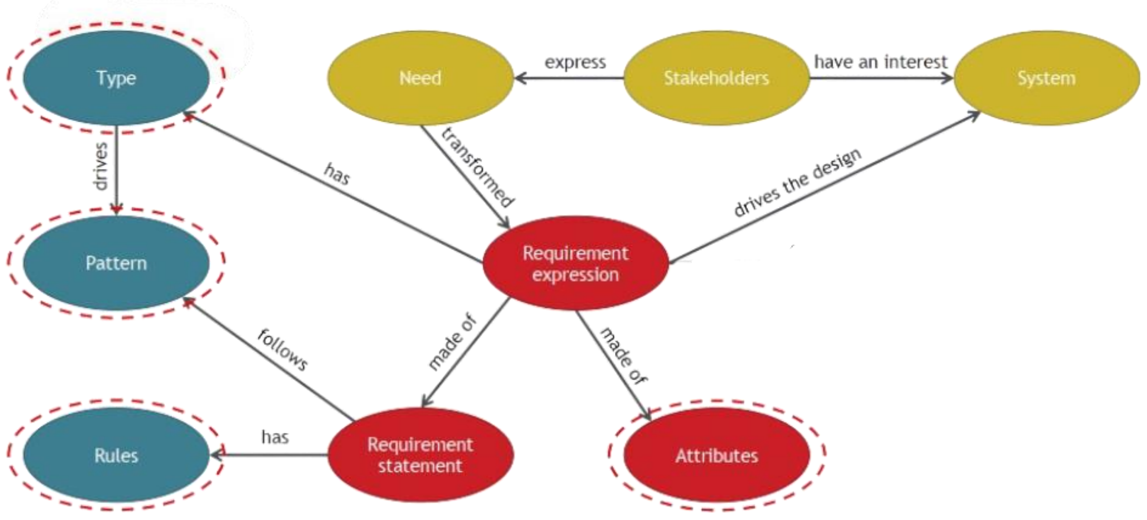


Figure 11: Requirement Statement structure(Ryan & Wheatcraft, 2022)

In agreement with the standard proposed every requirement expression has to satisfy the following characteristics(Ryan & Wheatcraft, 2022):

- It has to be necessary, it must define a capacity, characteristic, or limit that is essential for the system. In the event the requirement expression is deleted the capability described by it cannot be expressed by the other existing requirements.
- It has to be unambiguous, it can only be interpreted in one way, with no possibility of misunderstanding its meaning.
- It has to be consistent, it cannot conflict with other requirements expressions.
- It has to be complete, it does not require further processing.
- It has to be traceable, it must be possible to be able to identify where a requirement comes from.

- It has to be feasible, it is to be achieved by implementing contemporary technologies.
- It has to be verifiable, the architecture of the system has to be able to satisfying the requirement expression.

These characteristics ensure the clarity, validity, and practicality of the requirements throughout the system design process.

Every requirement statement is characterized by an attribute type. The attribute type helps to categorize the requirement, examples of attribute types are functional, performance, design constraints, and so forth. The attribute type is chosen based on the requirement statement meaning and the requirement statement structure depends by the attribute type(Boggero et al., 2020). Depending on the meaning of the requirement the designer selects a specific attribute type and then based on it the designer has to follow specific patterns, an example of patterns is provided in Table 2. The employment of these patterns serves to ensure the requirement statement is unambiguous. In addition, many linguistic rules have to be observed by the requirement statement, some of these are show in Table 1.

Rules N.	Rules Text
ACTIVE VOICE	Use the active voice in the need statement or requirement statement with the responsible entity clearly identified as the subject of the sentence.
DEFINED TERMS	Define all terms used within the need statement and requirement statement within an associated glossary and/or data dictionary.
DEFINITE ARTICLES	Use the definite article "the" rather than the indefinite article "a".
COMMON UNITS OF MEASURE	When stating quantities, all numbers should have appropriate and consistent units of measure explicitly stated using a common measurement system in terms of the thing the number refers.
VAGUE TERMS	Avoid the use of vague terms
SUPERFLUOUS INFINITIVES	Avoid the use of superfluous infinitives such as "to be designed to", "to be able to", "to be capable of", "to enable", "to allow".

Table 1: Example of linguistic rules that requirement statement has to follow

Attribute Type	Requirement Patterns	Example
Functional	The SYSTEM shall [exhibit] FUNCTION [while in CONDITION]	The aircraft shall provide propulsive power [during the entire mission]
Performance	The SYSTEM shall FUNCTION with PERFORMANCE [and TIMING upon EVENT TRIGGER] while in CONDITION	The aircraft shall fly with a speed of minimum Mach 0.8 while in cruise
Design (Constraint)	The SYSTEM shall [exhibit] DESIGN CONSTRAINTS [in accordance with PERFORMANCE while in CONDITION]	The aircraft shall have technologies with maturity TRL 9 [in 2020]
Environmental	The SYSTEM shall [exhibit] CHARACTERISTIC during/after exposure to ENVIRONMENT [for EXPOSURE DURATION]	The aircraft shall be maneuverable in case of icing conditions [for the entire flight]
Suitability	The SYSTEM shall exhibit CHARACTERISTIC with PERFORMANCE while CONDITION [for CONDITION DURATION]	The aircraft shall exhibit a steady gradient of climb of minimum 2.4% while the condition of one-engine-inoperative

Table 2: Example of attribute types and their patterns

Attributes, at last, complete the requirement expression. They guarantee traceability, identification, uniqueness and the verifiability of the requirement. The following Table 3 shows the attributes used.

Requirement's attibrutes:
ID
Type Attribute
Need Source
Stakeholders
Version
Verification
Priority
Means of Compliance
Author

Table 3: List of used requirements's attributes

After the translation of needs into requirement expression, the possibility exists to add further requirements derived from the existing ones(Ryan & Wheatcraft, 2022). These additional requirements delineate a specific component of the stakeholder's requirement or restrict a subsystem of the primary system of interest. The generation of requirements derived from other requirements creates a hierarchy that enables traceability from the stakeholder requirement to the component requirement and vice versa.

3.4 System Architecture

Once the stakeholder and system requirements have been determined, the system architecture design phase can begin, Figure 12. The ultimate goal of this phase is to generate a system architecture that is able to satisfy the stakeholder requirements.

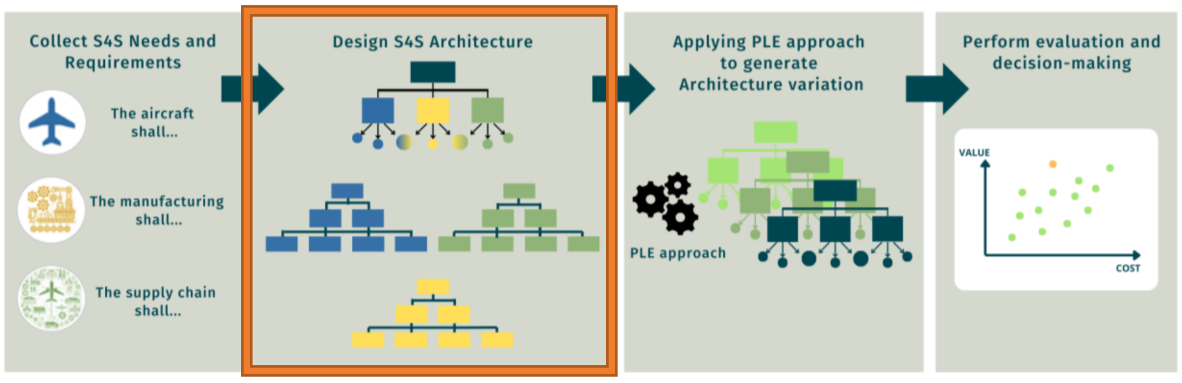

Collect S4S Needs and Requirements
The aircraft shall...
The manufacturing shall...
The supply chain shall...
Design S4S Architecture
Applying PLE approach to generate Architecture variation
PLE approach
Perform evaluation and decision-making
VALUE
COST

Figure 12: Second methodology step

To achieve this objective, it is considered good practice to follow specific steps(A Practical Guide to SysML The Systems Modeling Language, n.d.; Weilkiens, n.d.):

1. Defining the system boundaries;
2. Identifying the core functions the system must perform;
3. Decomposing the complex system into subsystems until reaching the component level.
4. Connect components and assign behavior

These steps are explained in detail in section 3.4.1.

Prior to proposing the methodology used for the system's architecture design, it is crucial to introduce the SysML language.

3.4.1 SysML Structure

SysML stands for **Systems Modeling Language**, it was first introduced in 2007 to replace the UML language, and since then it became a fundamental tool in the development of systems.

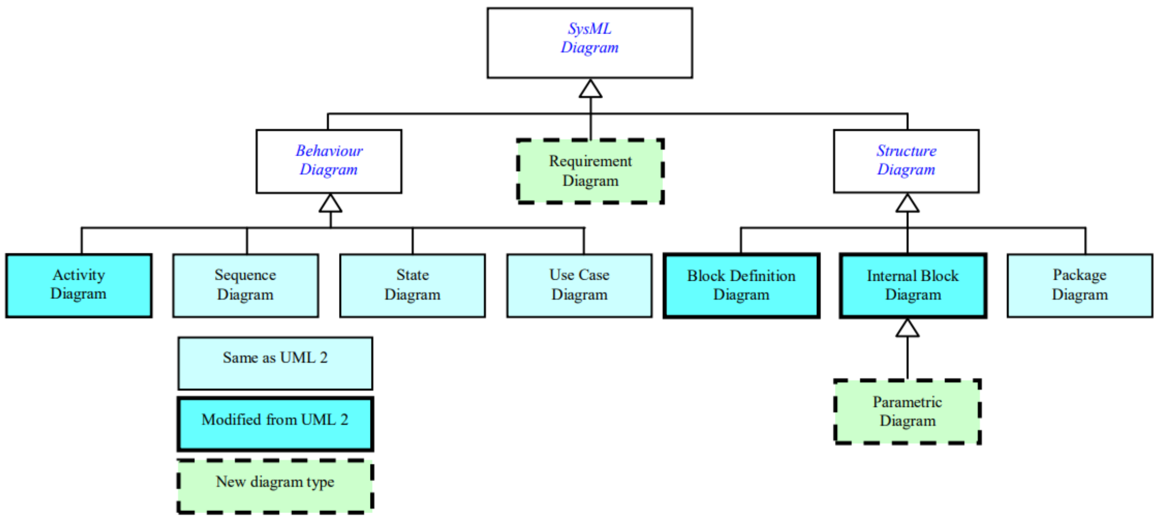


Figure 13: SysML structure (Benina et al., 2023)

Figure 13 above shows the structure of the language. SysML is composed of nine diagrams, highlighted by the colors in the picture, each diagram is intended to represent a specific view of the system. But in general, there are three functions that each diagram can be associated with(Friedenthal, 2014; Weilkiens, n.d.):

- Structural
- Behavior
- Requirement structure

Starting from the requirement diagram, was introduced to enable the possibility of showing requirements inside the system model. Usually, requirements are managed using specific software outside of the system model, the requirement diagram permits to import and show these requirements. On top of that, it has the ability to represent the dependence between the requirements and other model elements. The two dependences <<*deriveReqt*>> and <<*satisfy*>> can be used to visualize, the derivation from requirements to others for the former and the satisfaction of a requirement by the design. Other dependencies are <<refine>> <<verify>> and <<rationale>>, in this work they are not used.

The structure of the system is expressed using four diagrams: Block Definition Diagram, Internal Block Diagram and Package Diagram.

The **Block Definition Diagram (BDD)** is used to express the relationships between blocks. The main relationship used is the <<*Directed Composition*>> that is used to express the hierarchy between blocks. In the example in Figure 18 can be seen that the Power train is composed by the other blocks under it, this connection is expressed via the reference introduced. The connector having an arrow at one end and a turbot on the other is the SysML symbol for a <<*Directed Composition*>>. The blocks connected to the Power train are parts of that in the **Internal Block Diagram (IBD)**. These diagram boundaries represent the contours of the depicted block, meaning that everything inside represents components or subsystems of it. The boundaries can have interconnection, those represent the interconnection with other subsystems that are not part of the one depicted by the internal block diagram. The elements used to represent the connections between the block's components are:

• Ports: standard ports are used to represent a connection request-reply between two elements, in contrast, flow ports are used to express a flow of data, materials, or else that can enter, leave, or both the part. If only one element is flowing the port depicted is an atomic flow port, if a set of elements are flowing it's necessary to model a "fuel specification", a list of all the items that flow. In this case, the port is a non-atomic flow port. The element flowing can be a block, a ValueType, a DataType, or Signal.

• Connectors: used to visualize the elements to which a component is connected. It can be characterized by flows. Flows represent transmitted elements or information (Item Flow) or events that trigger a state in another component.

• Parts: These represent the system components. Parts cannot be created independently, they are automatically generated when a system is decomposed in the BDD. The blocks that compose the father block are represented as parts in its IBD. This assists the designer in ensuring that no parts are defined within an IBD unless they have already been assigned to a system.

Figure 19 in Section 3.4 provides an example of an Internal Block Diagram.

The **Parametric Diagram** is a specialized IBD, it is stripped of some elements and a new element class of block is introduced to represent constraints that have to be applied to system properties. This block is the *constraintBlock.* This block supports the engineer in representing properties' constraints. A constraint can also be a mathematical formula. Parameters and properties are depicted by small square boxes, while the *constraintBlock* is depicted by a box.

Finally, the **Package Diagram** is used to organize the model, the model can be organized in packages (or folders), they are then shown in the package diagram.

The behavior of a system can be designed using the remaining diagrams (in Figure 13 the four diagrams on the left). The **Use Case Diagram** describes the behavior in terms of high-level functionality. It is composed of three elements: Actors, System Context, and Use Cases.

- Actors represent all entities that interact with the system, they can be individuals or other systems but are always external to the System Context.
- The System Context is represented by a box and the functions that the system must perform are represented by the function inside this box, outside are represented by the actors and the systems that interact with the system of interest.
- Use Cases, describe interactions between the system and an actor, and usually the interaction is initiated by the actor. They can be considered as the fundamental functions that the system has to complete derived by the requirement expression.

Activity Diagram provide a more detailed representation of the actions that the system must perform and the sequence in which they should be executed. An activity describes flows that consist of various actions and additionally specifies input and output data that may be required during an action flow. The purpose of the diagram is to illustrate the order of actions, whether they can be executed in parallel, and to indicate the requirements. The diagram always commences with an "initial flow" that initiates the control flow. Two different kinds of flows are used in this diagram: control and object flow. The first is used to indicate a sequence of actions but no physical object is flowing between the two actions, the latter is used to show the flow of materials, information, data, etc. activity's flow concludes when the final node is reached. The **Sequence Diagrams** is used to represent the sequential exchange of messages between actors, systems, and the system's subsystems. These diagrams can also be employed to describe interactions between system components. The diagram represents a timeline which is represented by element lifelines, messages, and responses, the last two are depicted by arrows connecting the various lifelines. Messages can be synchronous or asynchronous depending on whether the sender expects a response from the receiver or not. When the receiver receives a message, it may indicate the need to initiate behavior or request information. The **State machine Diagram** is used to model the different states of the system and their transitions.

3.4.2 S4S Design Methodology

The first step of the system design phase is the identification of system boundaries. This operation is crucial to the system's success, a lack of clarity in defining these boundaries can lead to an oversizing of the system(A Practical Guide to SysML The Systems Modeling Language, n.d.; Weilkiens, n.d.), the modeling of external functions, or the failure to identify critical functions. For a system such as the one under examination, this activity may be particularly complex due to both the scale and the intricacy of integrating the design of three complex systems simultaneously. To delineate the system effectively it is necessary to combine the designer's intuition with a systematic approach.

The first step is to consider the definition assigned to the system, and then, use this definition and a degree of intuition to establish preliminary and approximate system boundaries. The next step involves identifying the elements that are part of the system, those that interact with it, and those that have no relationship with it. In the early stages, useful criteria for determining whether an element belongs to the system is to verify if the designer has control over its design or not the following example illustrates this concept.

Suppose you want to design a two-seater car for road use. By common knowledge or intuition, it is easy to imagine the following things: the vehicle must interact with the driver, the passenger, and the road. It is easy to understand that the road is connected to the system of interest but is not part of it the vehicle designer cannot modify the road, he can only consider the possible scenarios and possible interconnections with the system. The same applies to the vehicle passenger and the driver. Therefore, while the road and the driver interact with the system, they are not considered part of it. A more rigorous approach to defining system boundaries involves utilizing the requirements identified. By analyzing stakeholder requirements, functional ones in particular, it is possible to derive the core functions that the system has to perform. By considering both the entities that perform the functions and those that interact with the system to be part of them, it is possible to further refine the system boundaries. As the design process progresses, it is necessary to iterate previous steps to continuously refine and adapt both the system's functions and its boundaries.

To facilitate the boundaries identification process, the SysML language offers a dedicated diagram already introduced, the Use Case Diagram.

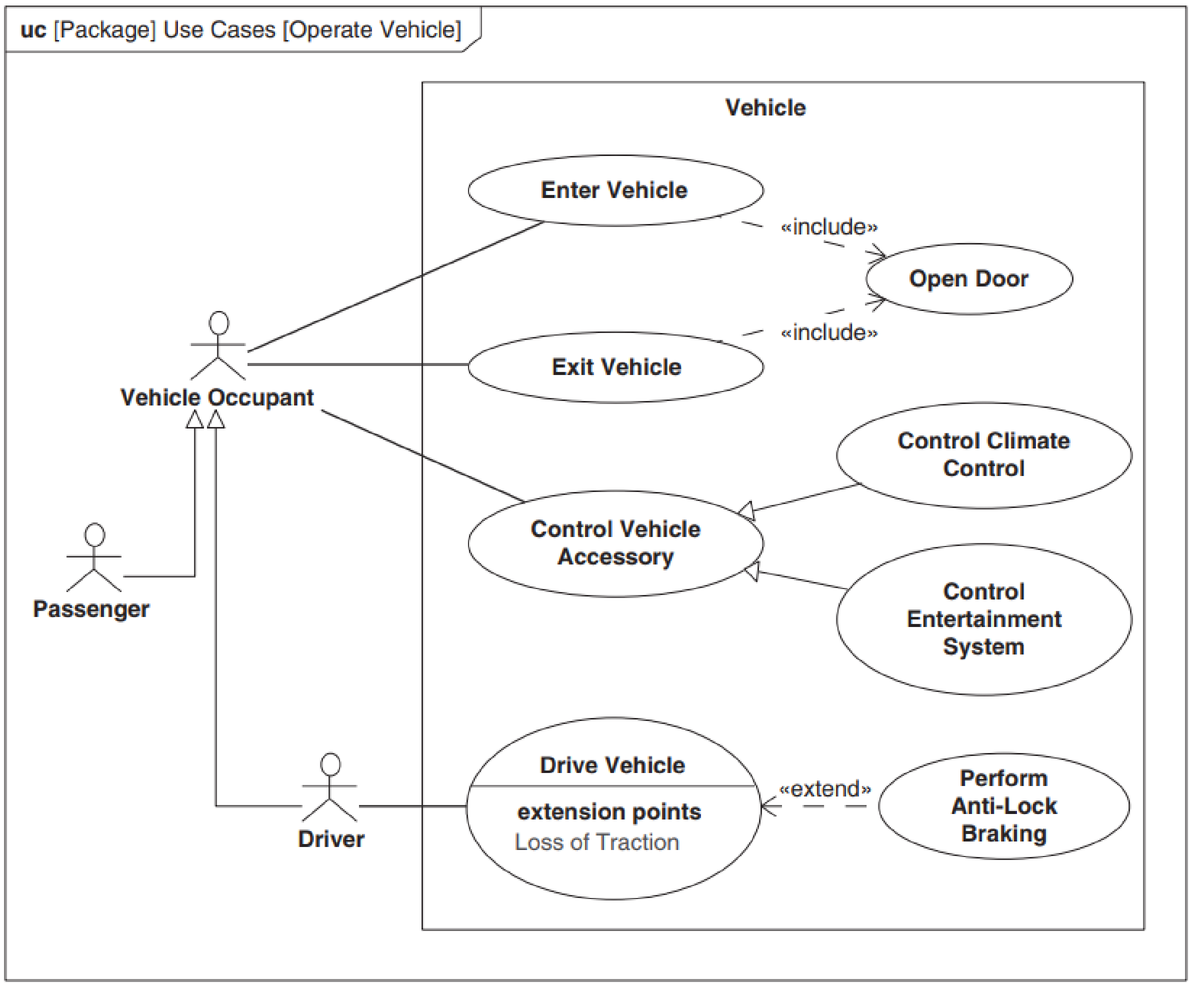


Figure 14: Example of a Use Case Diagram(Friedenthal, 2014)

As previously stated, the functions introduced are of a high level and therefore require further definition. It is important to remember that the requirements drive the modeling of functions.

The relationship between the utilized requirements and high-level functions must be demonstrated to ensure traceability throughout the project. For this reason, a satisfying relationship is necessary. The child requirements of functional ones are employed to further specify the use cases. As previously mentioned, high-level requirements are typically associated with high-level functions, while derived requirements correspond to increasingly specific activities. Use cases are specified in detail by decomposing the actions required to perform the functions. To derive these functions, it is always necessary to refer to the requirements, as they guide the designer. Additionally, the designer must possess knowledge of the subject under analysis; therefore, a literature study or consultation is required in parallel.

As stated, Activity Diagram and Sequence Diagram are the tools that a system designer uses to specify the high-level function. Having the use case and the actor, the sequence diagram is usually the one that is to be analyzed first. Considering the car example and its Use Case diagram, the Drive Vehicle use case is taken into consideration. From the diagram, the only actor interacting with the use case (high-level function) is the driver. The two entities in connection are the driver and the vehicle. The diagram is represented in Figure 15, in this diagram, a series of behaviors are listed "*turn on vehicle*", "*control power*", "control direction" and so on. All those behaviors are derived from the requirements. For example, "control direction" can derive from a functional requirement that requires that *the vehicle shall be able to change direction,* this requirement can be the child of a more general requirement *the vehicle shall be able to be driven.* This is repeated in cascade. To show an easy example of messages between components the "*turn on vehicle*" sequence is presented in Figure 16

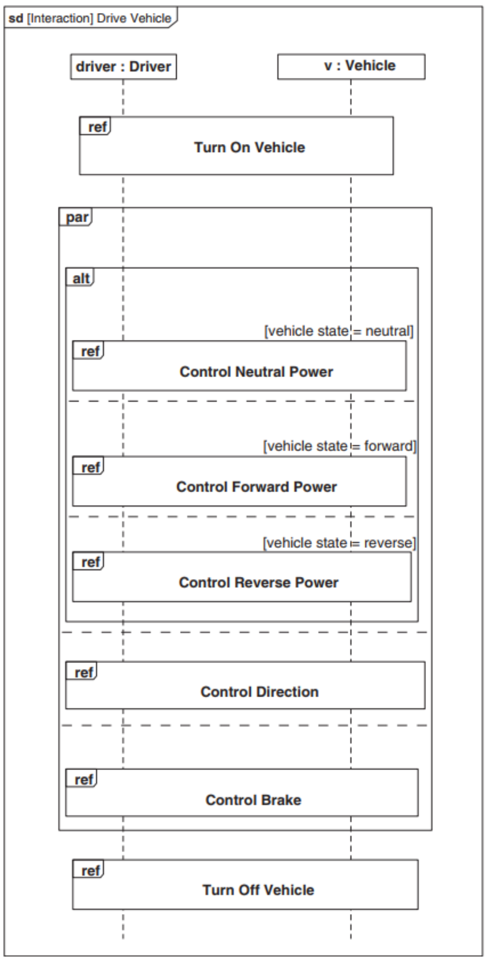


Figure 15: Example of a use case specification using a sequence diagram(Friedenthal, 2014)

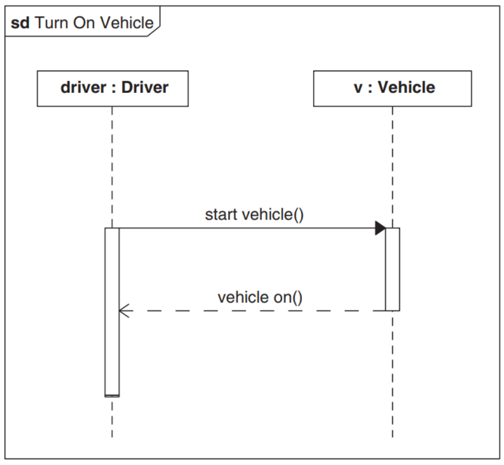


Figure 16: Example of Sequence Diagram(Friedenthal, 2014)

The driver sends a message "start vehicle" and the system replies with a message "vehicle on". Another important piece of information that can be modeled in this diagram is time, it goes from the top to the bottom. The two bars represent the time the elements are active. Activity diagrams can also express system behavior and specify the functions. Usually, it is used when representing continuous behaviors. The same link activity to the requirement introduced can be applied here. Note that not every action has to be connected to a requirement, same actions are introduced based on system engineer knowledge or literature studies. Figure 17 represents the ability to control power, the driver has to perform two actions "*control accelerator position*" and "*Control Gear Select*" The system receives as input two object flows "*accelerator command*" and the "*gear selected*" to fulfill the "*provide power*" action. The activity ends when the signal "*ignition off*" is sent by another activity.

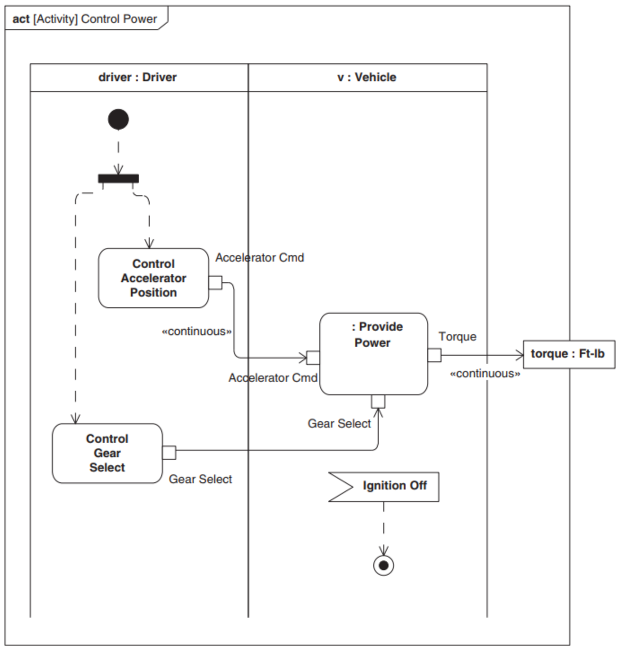


Figure 17: Activity Diagram example(Friedenthal, 2014)

Use Cases, Activity Diagrams and Sequence Diagrams serve the purpose of describing how the system must behave, as required by the project requirements. This group of diagrams, along with the State Machine Diagram (not utilized in this project), considers the system as a black box at this stage and enables the designer to define what is referred to as the system's behavior. Once this behavior is established, the component design phase can commence.

The component design phase marks the beginning of the actual modeling of the system architecture. At this stage, the designer perspective switches from a black box perspective to a white box. The identification of components remains linked to the system's activities and functions and the ultimate objective is to integrate components that fulfil the requirements and all together are able to perform the function asked for by the stakeholder requirements. The same approach utilized to derive behaviors from the requirements is adopted to derive components from those behaviors and the performance requirement.

The decomposition of the system follows a hierarchical order, it is broken down into N subsystems, which are divided into N sub-subsystems until the components are identified. The collective assembly of components must ensure the fulfillment of both the functions derived and the system requirements, including performance requirements and constraints. In SysML, the system breaks down in a hierarchical structure and can be facilitated through the use of a Block

Definition Diagram (BDD). To accomplish the action "*provide power*" in Figure 17 some components are required. The selection of the components derives from literature studies or experience. Taking into consideration all the behaviors of the system, the decomposition of the element can commence. In Figure 18 the example for the car, the action stated before is performed by the *power train* block. Every component of the sub-system will perform one or more behaviors.

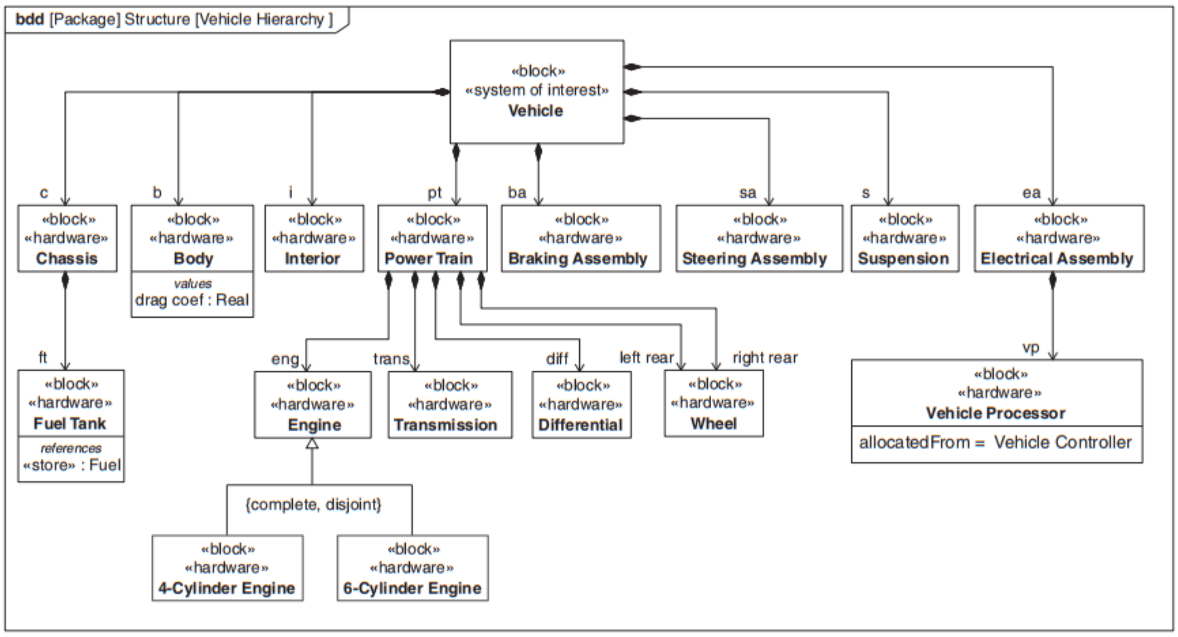

bdd [Package] Structure [Vehicle Hierarchy]
«block» «system of interest» Vehicle
c
b
i
pt
ba
sa
s
ea
«block» «hardware» Chassis
«block» «hardware» Body
values
drag coef : Real
«block» «hardware» Interior
«block» «hardware» Power Train
«block» «hardware» Braking Assembly
«block» «hardware» Steering Assembly
«block» «hardware» Suspension
«block» «hardware» Electrical Assembly
ft
«block» «hardware» Fuel Tank
references
«store» : Fuel
eng
trans
diff
left rear
right rear
vp
«block» «hardware» Engine
«block» «hardware» Transmission
«block» «hardware» Differential
«block» «hardware» Wheel
«block» «hardware» Vehicle Processor
allocatedFrom = Vehicle Controller
{complete, disjoint}
«block» «hardware» 4-Cylinder Engine
«block» «hardware» 6-Cylinder Engine

Figure 18: Example of a hierarchical structure in a BDD(Friedenthal, 2014)

The hierarchical decomposition shown in the BDD is still not capable of showing how each subsystem operates for example, while the Power Train composition is clear, how the four elements can interact together to fulfill the "*provide power*" function is still unclear. These components interact by exchanging materials information or torque and interfacing with one another. To model the internal exchange of information within a system, the SysML language provides the Internal Block Diagram. Figure 19 represents the IBD of the car's power subsystem, and the red rectangle represents the IBD of the Power Train component.

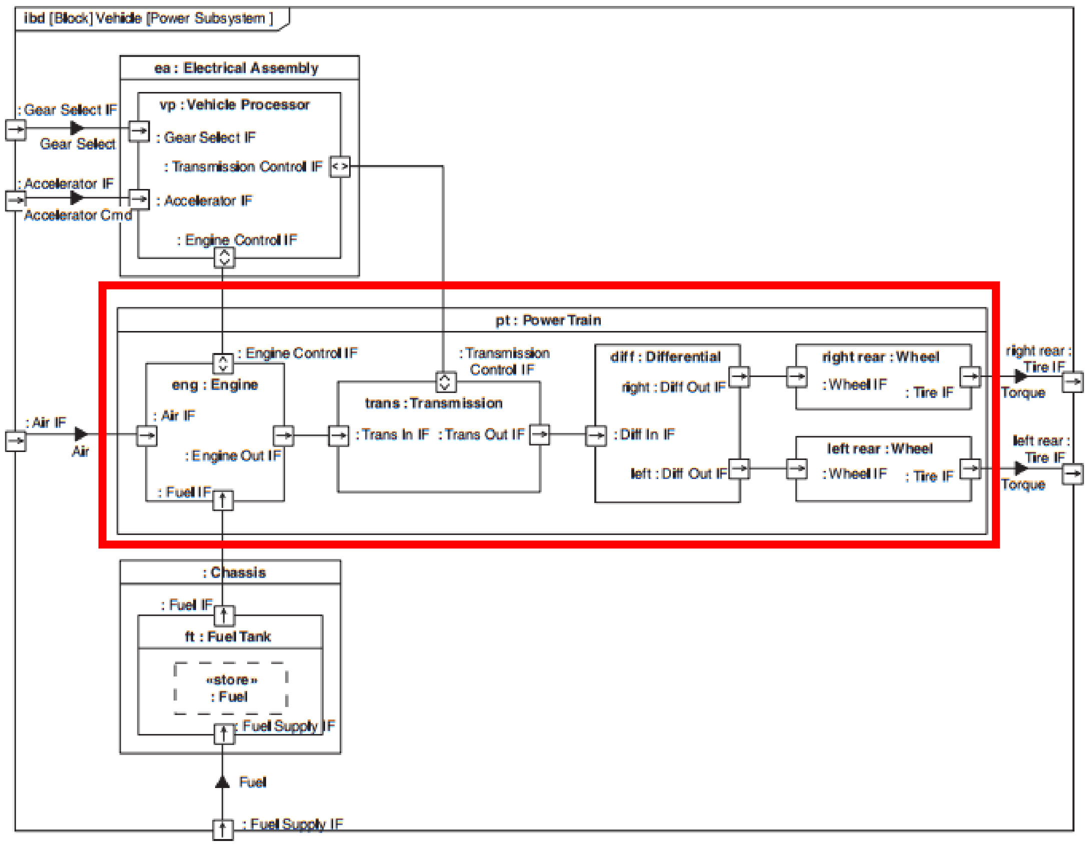


Figure 19: Example of IBD and its structure(Friedenthal, 2014)

Using the elements (ports, connectors, itemflows, parts etc) presented in Chapter 3.4.1 the interaction between the components is shown. Port kind and number and itemflows choice rely on the designer of the system. The same system can have different configurations depending on who the designer is. The procedure presented for the power train has to be repeated for every component of the system, until all the components are designed or until the level of detail required is reached.

3.5 System variability

The outcome of the second step is a fundamental structure that encompasses the three systems. Now the objective shift is to develop a family of variants.

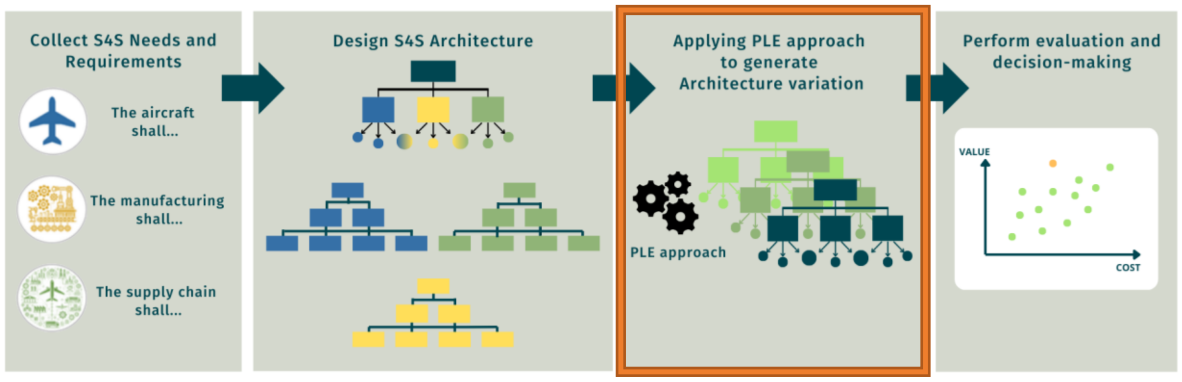

Collect S4S Needs and Requirements
The aircraft shall...
The manufacturing shall...
The supply chain shall...
Design S4S Architecture
Applying PLE approach to generate Architecture variation
PLE approach
Perform evaluation and decision-making
VALUE
COST

Figure 20: Third methodology step

Exploring various architectures and solutions of the system, during the conceptual phase, is important because the decisions made during this phase, while not incurring immediate material costs, have significant downstream impacts on the production process. Decisions made at this stage commit substantial financial resources later in the process. This is the reason why, via the generation of variants, it is possible to enhance system knowledge, thereby enabling a more informed selection of the optimal solution.

To develop the variants, it is necessary to introduce all possible design solutions. Two choices can be made, create a large number of systems in parallel, or, as this study proposes, leverage a technique commonly used for generating product families, Product Line Engineering (PLE). The nature of the traditional approach leads to spending more time or using more staff. Using the latter approach to generate different variants from a single model can result in cost and time savings. But a shift from the traditional PLE approach has to be made, rather than working on a plethora of architectures of similar systems, identifying shared elements, and selecting the feature from a catalog to generate a family of systems, the approach proposed uses fixed elements, such as requirements, functions and some components and a feature model of elements can be changed to generate a variant of the same system.

A small introduction to PLE is presented below.

NOTE: Usually, in PLE terms used are "Product Line" and "Product"; however, these terms can be translated as "system family" and "system" and in this work the last ones will be used.

The four typical processes are as follows:

- **Define Product Line Feature Models (or Feature Catalogue):** Features represent the variability of member products derived from the product line. They are defined as distinguishing characteristics of a product that are shared with and understood by customers (ISO/IEC 26580:2021, n.d.). Features are modeled hierarchically and can be interconnected through requires and conflict relationships (Forlingieri & Weilkiens, 2022a). Mandatory features are also allowed, primarily when they are parents of optional features.
- **Define Product Line MBSE Assets (or Shared Asset Supersets):** To generate a system, it is necessary to develop and model all the required assets (e.g., requirements, architecture models, V&V). In a family system, it is important to distinguish between common assets (shared across systems) and varying ones (specific to individual systems). These assets are linked to system features to systematically manage variability(Forlingieri & Weilkiens, 2022a).
- **Select Member Product Feature Configuration (or Bill-of-features):** In this phase, by selecting the desired features it is possible to create a feature configuration for different systems(Forlingieri & Weilkiens, 2022a).
- **Derive Member Product MBSE Assets (or Product Asset Instances):** In this activity, feature configurations are translated into concrete, customized MBSE models tailored to individual systems. The system's components are filtered based on the selected feature configuration, thereby generating the system(Forlingieri & Weilkiens, 2022a).

As stated, the proposed approach differs from the traditional one. It will use the PLE ability to easily generate different architectures from an initial set of elements, but rather than create a product line (or system family), the objective is to identify a family of variants of the same system. So, the activities list mentioned above is distinct for the approach used.

The feature catalog continues to represent which features can be selected, but there is no longer a shared super asset because the assets belong to a basic architecture, the distinction is no longer between shared and non-shared assets, but rather between varying and stable assets. The third activity remains unchanged, but the selection of features determines which architecture is generated. Finally, models of various architectures are created, resulting not in a family of systems but in a family of architectures for the same system. This approach is applied to the three systems: the aircraft and the two enabling systems, supply chain and manufacturing.

The PLE approach involves modifying the baseline architecture by introducing variability points(Hummell & Hause, 2015), i.e. specific locations where the system can accommodate different design solutions. The variation points are in the system architecture, so neither the system requirements nor its core functions will change, the variability will be limited to the components and their interconnections. Adding variation points and new elements in the system model leads to the creation of a 150% model, this model is so named because it considers all possible components, both fixed and variable. Through a selection of variability points a specific system variant is generated resulting in a 100% system, meaning a fully defined and realizable configuration.

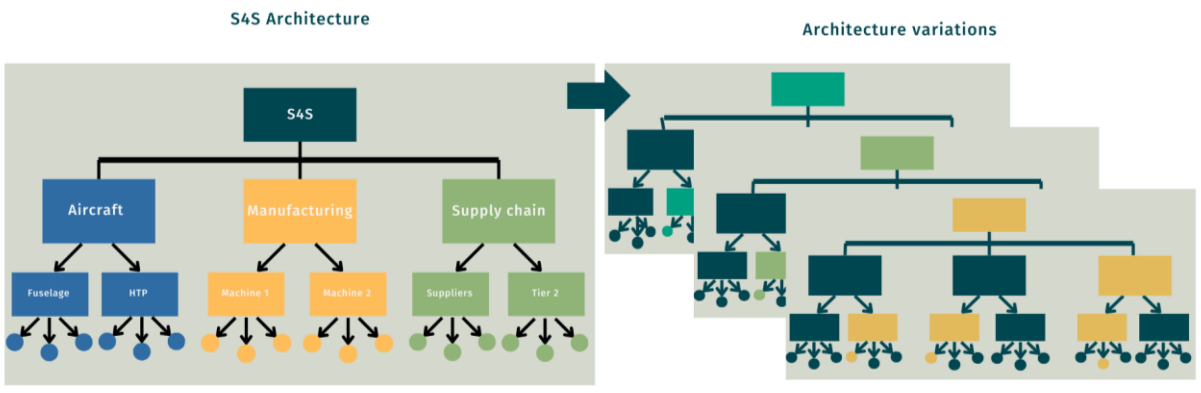


Figure 21 provides a visual representation of the explained concept. The box on the left represents the 150% model. Machine 1, which is part of the manufacturing system, has three possible variants, which means that it can have three different structures, the same components can be connected in different ways, or can perform the same process with different performance. In the three generated variants (box on the right), each includes only one of the three alternatives, resulting in a fully defined and operational final model.

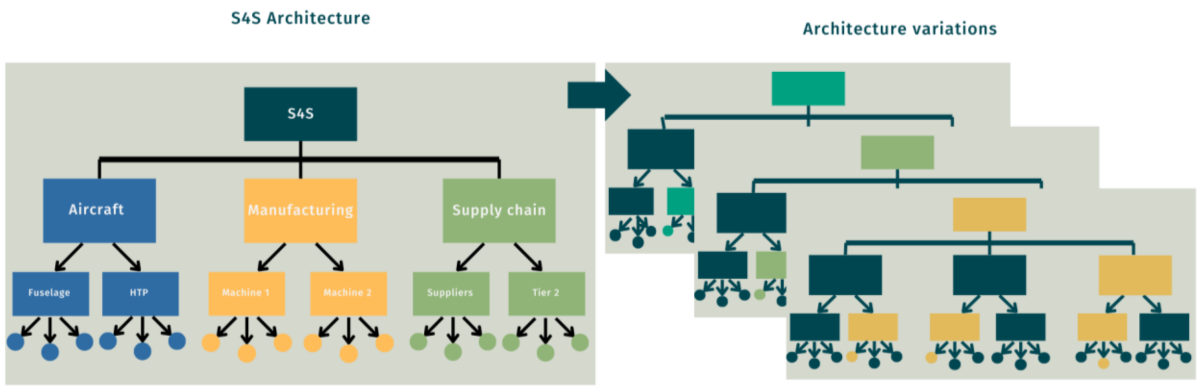


Figure 21: Representation of PLE approach of a system model

Given their different compositions, these newly generated architectures will exhibit varying overall system performance. Before selecting the optimal configuration, a thorough analysis of these performance differences must be conducted.

In terms of best practice, the first step is to identify the points that may vary. The identification of these points is dependent on the intended design, and in the case of the examined test, variability points are identified in processes, facilities, and materials. These points are sought within the identified base architecture, once the variability points have been determined the model is populated with additional blocks representing the variables to be introduced. Facilities are incorporated into the supply chain, possible processes are added, and potential materials are introduced. By adding these elements, the base model is enriched with components that will not be present in all variations, and, at the end of this phase, the model becomes the 150% model.

The selection is then entrusted to an external configurator, which is used to select the components to be retained in a particular architectural variant. The configurator must enable the implementation of rules to guide the selection of features, as without such rules, there is a risk of selecting conflicting elements. Rules or relationships are modeled inside the feature model, two features can be connected by a <<*conflict*>> relationship or <<*requires*>>, the first makes sure that if one feature is selected another one is excluded, the second, if one feature is selected others, are selected automatically(pure-systems GmbH, 2024).

These rules between the various features are illustrated in Figure 22. In the example in the picture, the "Skin Aluminum Machining" feature requires that the material selected is Aluminum (Skin Aluminum feature), if selected all the features listed cannot be selected anymore.

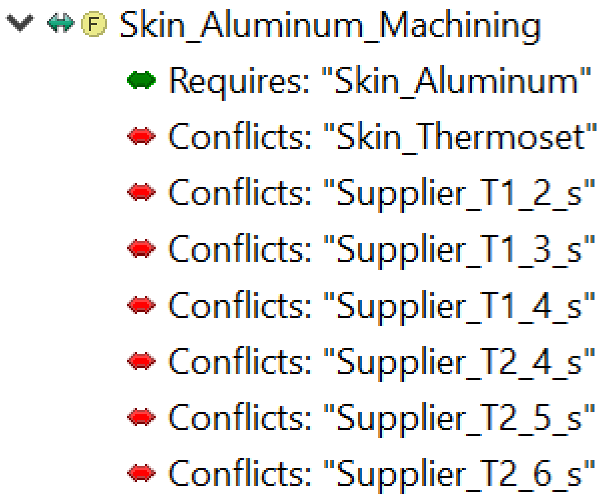
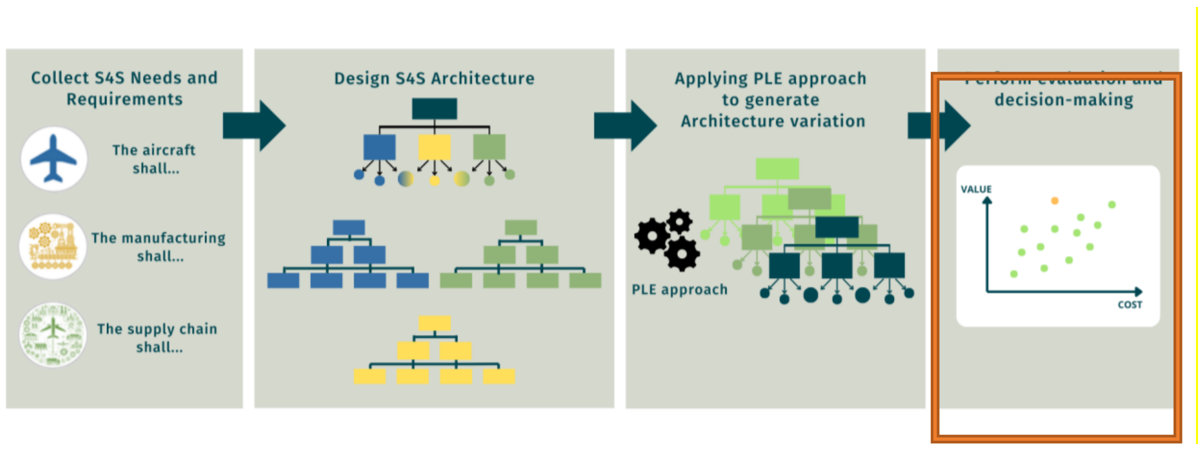

Skin_Aluminum_Machining
Requires: "Skin_Aluminum"
Conflicts: "Skin_Thermoset"
Conflicts: "Supplier_T1_2_s"
Conflicts: "Supplier_T1_3_s"
Conflicts: "Supplier_T1_4_s"
Conflicts: "Supplier_T2_4_s"
Conflicts: "Supplier_T2_5_s"
Conflicts: "Supplier_T2_6_s"

Figure 22:Example of relations implemented in the feature model

Finally, the features implemented in the selector are linked to the 150% model and the components in the system architecture. Through the selector, the features desired are chosen, the outcome is a variant of the system architecture. This process is repeated for all the possible variants until the ultimate result is a family of variants of the same system, which can then be analyzed.

3.6 Value-Driven analysis

This family of architectures is systematically evaluated to facilitate a comprehensive comparison among all architectural variations, to identify the one that best meets the stakeholders' requirements.



Collect S4S Needs and Requirements
The aircraft shall...
The manufacturing shall...
The supply chain shall...
Design S4S Architecture
Applying PLE approach to generate Architecture variation
PLE approach
decision-making
VALUE
COST

Figure 23: Fourth methodology step

The process of selecting the most appropriate alternative is influenced by various criteria defined by the stakeholders, which can include factors such as production cost, associated risks, and aircraft performance. In this study, Multi-Attribute Utility Theory (MAUT) is employed as the value-model technique to assess and identify the best alternative.

To carry out this evaluation, the methodology proposed in Donelli Giuseppa's PhD thesis (Donelli Giuseppa, 2024)is carried out. This methodology allows for the evaluation of systems that consider both production and product parameters.

For the case considered, the product performance depends on the material used during the manufacturing and the process. These two factors lead to different technical performances, like mass, that can have an impact on the performance of all systems, like the overall fuel consumption. The production line's performance is evaluated based on risk, quality, cost, and time. Each facility possesses a level of skill for a specific process, and the level of competence dictates the performance, to consider the performance of the entire supply chain all the facilities selected for that specific architecture variant must be taken into account. The MAUT value model usage enables the concurrent coupling of these systems(Donelli, Boggero, et al., 2023b)(Donelli, Ciampa, et al., 2023b). The criteria considered for the production domain have already been stated, the product criteria is fuel consumption (linked to the HTP's mass). All these criteria, except for cost, are aggregated into a final Value. To all performance indicators, a specific weight and a corresponding utility function are assigned. The weights reflect the relative importance of each criterion, while the utility functions quantify the stakeholders' preferences concerning each individual criterion (Ross, Rhodes, & Fitzgerald, 2005). Performance metrics are scaled and then aggregated into a single dimensionless term, referred to as Value. This Value is plotted as a function of cost, with the final graph illustrating the relationship between the value of each variant and its associated cost. This graph represents the solution trade space. The chosen solution is not necessarily the one with the highest Value or the lowest cost but rather depends on stakeholder preferences. The analysis provides a foundation for stakeholders to conduct a trade-off assessment, ultimately leading to the selection of the most suitable solution.

The figure presents two graphs. The graph on the left represents one of the metrics used to calculate the final system value (quality), while the graph on the right illustrates the total value-cost of the different variants. In this graph, the solution with the highest value achieves the best overall performance. However, it should be noted that the best-performing solution is not necessarily the one selected. Depending on stakeholder preferences or market demands, the chosen solution may prioritize lower cost, higher quality, or a balanced trade-off between these factors.

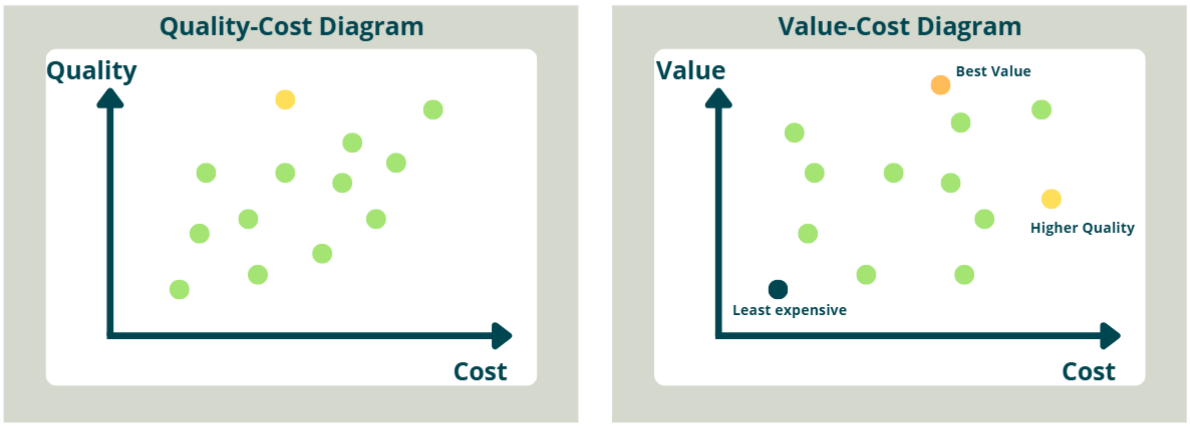


Figure 24: Example of results of a MUAT analysis

The equations used to evaluate the performance of the production system are presented below. Three contributors are presented in the equation, a fixed term, manufacturing term linked to the competence of the supply chain to perform the processes selected and a transportation term.

$$Cost_{SC} = Cost_{fixed} + Cost_{manufacturing} + Cost_{transportation}$$

$$Time_{SC} = Time_{fixed} + Time_{manufacturing} + Time_{transportation}$$

$$Risk_{SC} = Risk_{fixed} + Risk_{manufacturing} + Risk_{transportation}$$

$$Quality_{SC} = Quality_{fixed} + Quality_{manufacturing}$$

Equation 1: Production's performance equations(Donelli Giuseppa, 2024)

The fixed term relates to the fixed production time, quality, cost, and risk of all supply chain. The second addend is linked to the assembly competencies of the selected processes, the higher is the competence higher is the quality, and lower are cost, time, and risk. The third addend is linked to the transportation system, how much it takes to get from A to B, cost and risk involved. Quality is not affected by transportation. To calculate the transportation term the four kinds of transportation systems are considered, by road, by railway by airplane and by boat.

The Value is calculated by the sum of these four criteria and the product criteria. The formula is presented.

$$\text{Value} = \sum_{i=1}^{N} \lambda_i U(X_i)$$

Equation 2: Value equation

In which N is the number of criteria, λ_i is the weight and the U(Xi) is the attribute utility function. (Donelli Giuseppa, 2024)

Through this methodology the alternative that satisfies the most the stakeholders' needs and expectations is identified, ensuring that the final system architecture is balanced between production performance, aircraft performance, and cost considerations. This approach helps in the decision-making process.

4 Methodology Implementation

The processes of the S4S framework, in conjunction with the methods outlined in Chapter 3 for achieving the concurrent conceptual design of aircraft, manufacturing, and supply chain systems, can be executed through either a document-based or a model-based approach. The document-based approach prevails as the predominant method employed in the industrial sector for the management of requirements and information, relying on the creation of textual documents, reports, and other materials to represent information at a given point in time. These documents are shared both within the company and with system stakeholders. While the functionality of this approach is not contested, it presents a critical limitation, due to the very nature of documents, they become outdated as soon as they are saved document approach has at least 3 major limitations:

- Updated documents do not automatically change other related documents, so they need to be updated manually. This process can lead to errors, omissions, or misunderstandings, especially when done by different people.
- Updated documents are shared via email or other communication channels. However, changes made by one person are not always visible to others working on different parts of the same document. This can lead to inconsistencies even within a single document.
- The continuous updating process results in a huge increase in the number of documents generated.

Although the use of shared cloud-based files has mitigated some issues mentioned, particularly those related to real-time modification by more users, the limitations of the document-based approach persist.

Model-based approach is an emerging method for managing a system throughout its lifecycle. It is often referred to as Model-Based Systems Engineering (MBSE). Rather than relying on documents, this approach is based on a shared model that is updated in real time therefore information is interconnected and represented within the model itself.

The key benefit of this approach is that all information is contained within a single, authoritative model serving as the primary source of truth. Any modifications made to the model are automatically updated across the system ensuring that all stakeholders have an up-to-date representation of the system and minimizing consistency errors caused by working with outdated documents. The model-based approach employs diagrams, schematics, and flowcharts to represent information providing a clear and easily interpretable depiction of the system making it particularly suited for representing large and complex models, such as those found in the aerospace sector. Furthermore, the interconnection of tools enabled by this approach enhances requirement traceability by linking requirements to system functions and components. Finally, it facilitates automation simulations and optimization analyses. So, it helps to reduce the overall time required for system validation.

This next section focuses on presenting the tools employed in the application case discussed in Chapter 5 leveraging MBSE. The picture below schematically presents the tools used for each step of the methodology.

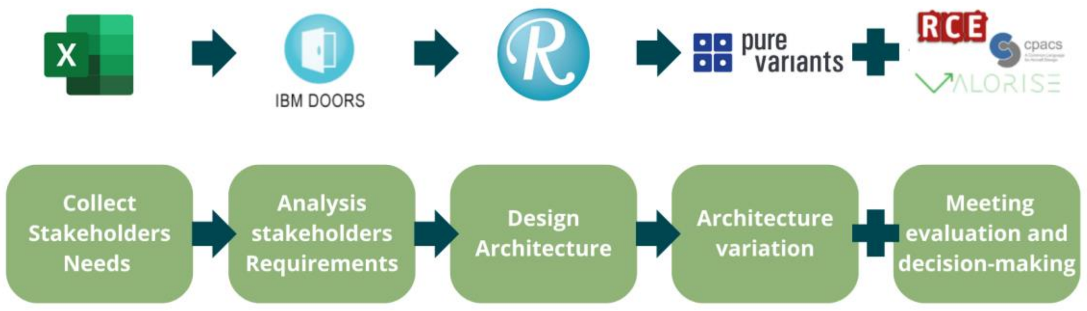

IBM DOORS
pure variants
RCE
cpacs
VALORISE
Collect Stakeholders Needs
Analysis stakeholders Requirements
Design Architecture
Architecture variation
Meeting evaluation and decision-making

Figure 25: Tools used in each step of the methodology(DLR, 2020; IBM, 2024b, 2024a; pure-systems GmbH, 2024)

4.1 Tools supporting system specification and architecting

As outlined in Figure 25, the implementation of these two phases is supported by a range of commercially available tools, these tools are not proprietary but rather consist of solutions currently on the market. Their integration is managed by an Environmental Workshop Manager, it facilitates seamless communication and mutual updating among them.

Stakeholders' needs are collected using an Excel file and subsequently imported into the web-based DOORS Next Generation (DNG) software by IBM(IBM, 2024a). Within this platform, requirements are directly defined from the stakeholders’ needs and managed. DNG facilitates efficient creation, organization, hierarchical structuring, and derivation of requirements statements. The software automatically generates a unique ID attribute for each requirement to prevent consistency errors, the remaining attributes have to be defined within the software itself by the user. The designers can specify the attributes that she/he find best. In the analyzed case study, the attributes used are those presented in Table 3 in Chapter 3.3.

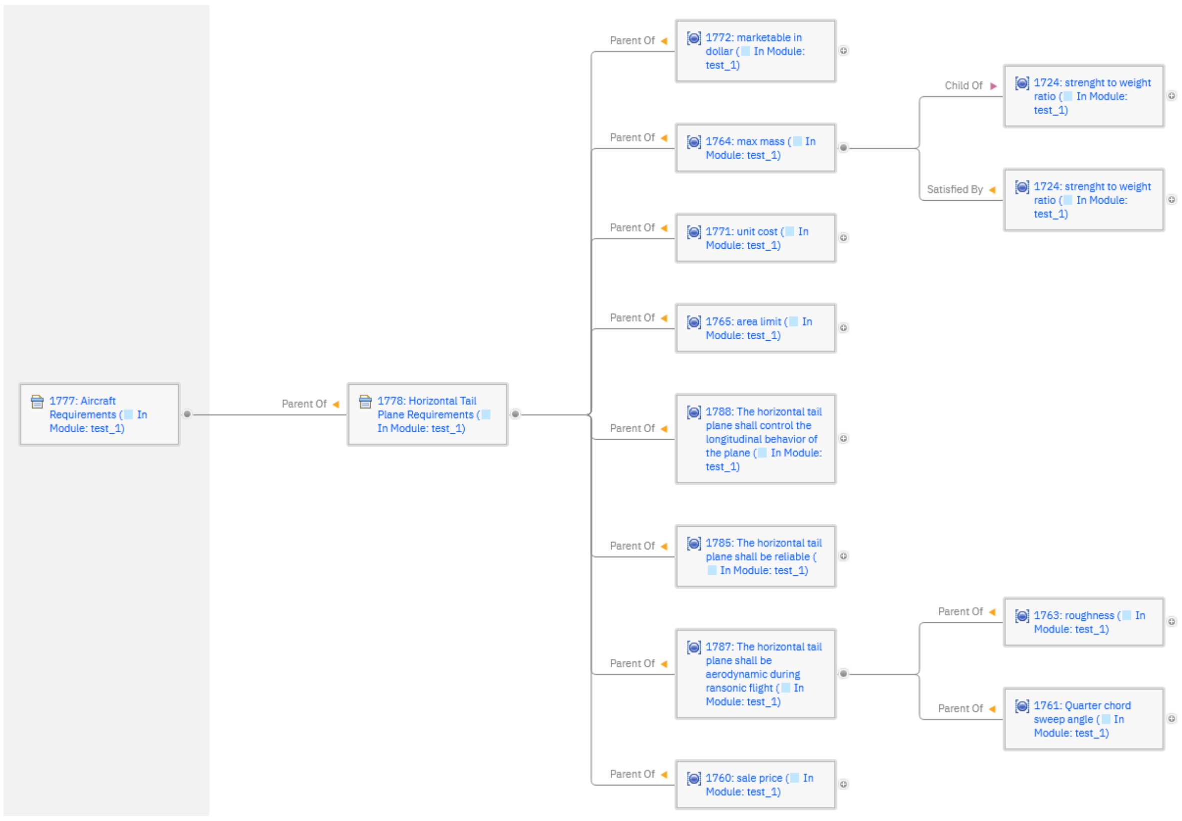

1777: Aircraft Requirements (In Module: test_1)
Parent Of
1778: Horizontal Tail Plane Requirements (In Module: test_1)
Parent Of
1772: marketable in dollar (In Module: test_1)
Parent Of
1764: max mass (In Module: test_1)
Child Of
1724: strenght to weight ratio (In Module: test_1)
Satisfied By
1724: strenght to weight ratio (In Module: test_1)
Parent Of
1771: unit cost (In Module: test_1)
Parent Of
1765: area limit (In Module: test_1)
Parent Of
1788: The horizontal tail plane shall control the longitudinal behavior of the plane (In Module: test_1)
Parent Of
1785: The horizontal tail plane shall be reliable (In Module: test_1)
Parent Of
1787: The horizontal tail plane shall be aerodynamic during ransonic flight (In Module: test_1)
Parent Of
1763: roughness (In Module: test_1)
Parent Of
1761: Quarter chord sweep angle (In Module: test_1)
Parent Of
1760: sale price (In Module: test_1)

Figure 26: Requirement tree inside IBM Doors Next Generation

Relationships between different requirements can be visualized directly within the software, in fact, DNG facilitates the generation of tree diagrams that illustrate these relationships, starting from one or more requirements, this feature enhance traceability and comprehension of dependencies. In Figure 26 an example of tree diagram is presented, the plus button close to the requirements is a sign that the tree can be further extended.

Additionally, requirement statements can be directly linked to system components, functions, or diagrams modeled in IBM Rhapsody through an OSLC connection.

The system architecture is developed using the IBM Rhapsody client(IBM, 2024b), which is based on SysML version 1. As previously introduced, IBM Rhapsody is integrated with DNG for requirement import and traceability. Rhapsody allows users to define the modelling language required for system architecture representation, and by selecting the appropriate language and specifying the user's level of expertise the software automates various operations, assisting designers in minimizing errors. Built-in guidance provides step-by-step instructions for correctly modeling a system, following the methodology introduced and SysML rules.

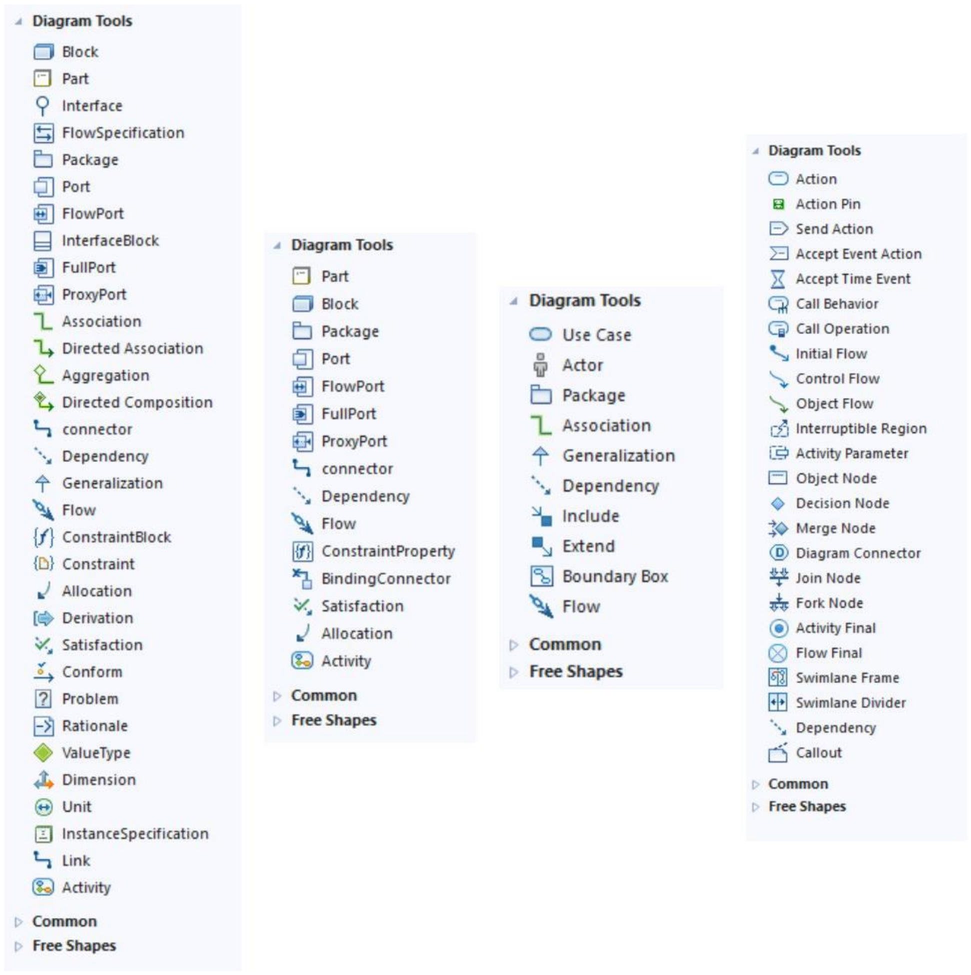
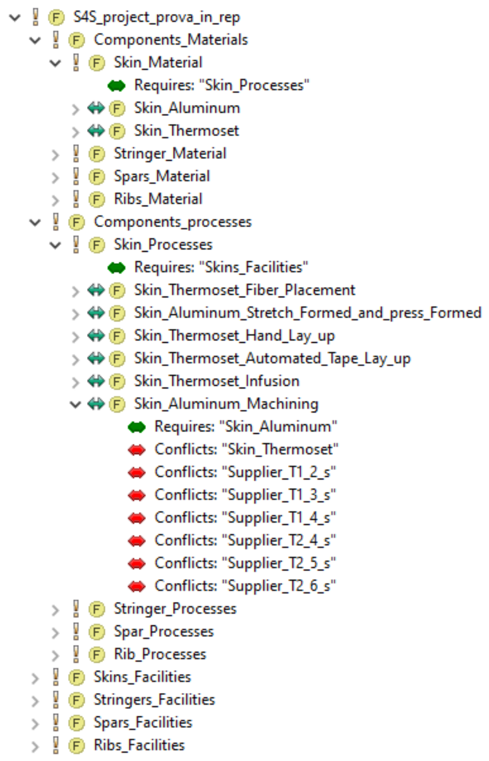


Figure 27: Example of how IBM Rhapsody menus change depending on the diagram in use. From left to right the menus presented are BDD, IBD, Use Case, Activity

IBM Rhapsody also supports the implementation of the Product Line Engineering (PLE) approach, the generated baseline architecture is expanded, using SysML formalism, to represent the 150% model. Through the Pure::Variants tool, variability can be applied to the system.

4.2 Tools supporting system variability

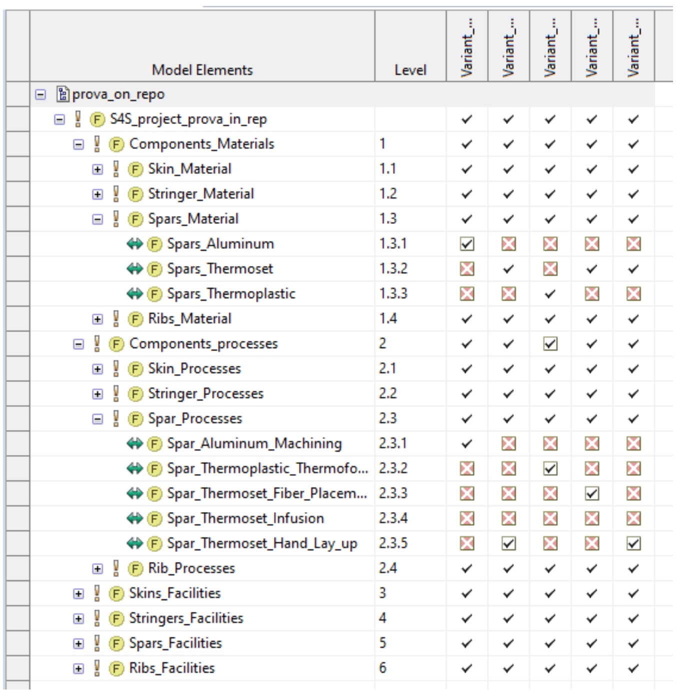
Pure::Variants is a software solution that facilitates the generation of a feature model(pure-systems GmbH, 2024). This client allows users to define relevant features and establish relationships among them, such as required, conflicts, and parent-child dependencies. In the feature model all the features of the system are represented.



```
S4S_project_prova_in_rep
  Components_Materials
    Skin_Material
      Requires: "Skin_Processes"
      Skin_Aluminum
      Skin_Thermoset
    Stringer_Material
    Spars_Material
    Ribs_Material
  Components_processes
    Skin_Processes
      Requires: "Skins_Facilities"
      Skin_Thermoset_Fiber_Placement
      Skin_Aluminum_Stretch_Formed_and_press_Formed
      Skin_Thermoset_Hand_Lay_up
      Skin_Thermoset_Automated_Tape_Lay_up
      Skin_Thermoset_Infusion
      Skin_Aluminum_Machining
        Requires: "Skin_Aluminum"
        Conflicts: "Skin_Thermoset"
        Conflicts: "Supplier_T1_2_s"
        Conflicts: "Supplier_T1_3_s"
        Conflicts: "Supplier_T1_4_s"
        Conflicts: "Supplier_T2_4_s"
        Conflicts: "Supplier_T2_5_s"
        Conflicts: "Supplier_T2_6_s"
    Stringer_Processes
    Spar_Processes
    Rib_Processes
  Skins_Facilities
  Stringers_Facilities
  Spars_Facilities
  Ribs_Facilities
```

Figure 28: Feature Model design in Pure::Variants, features, and relationships are displayed.

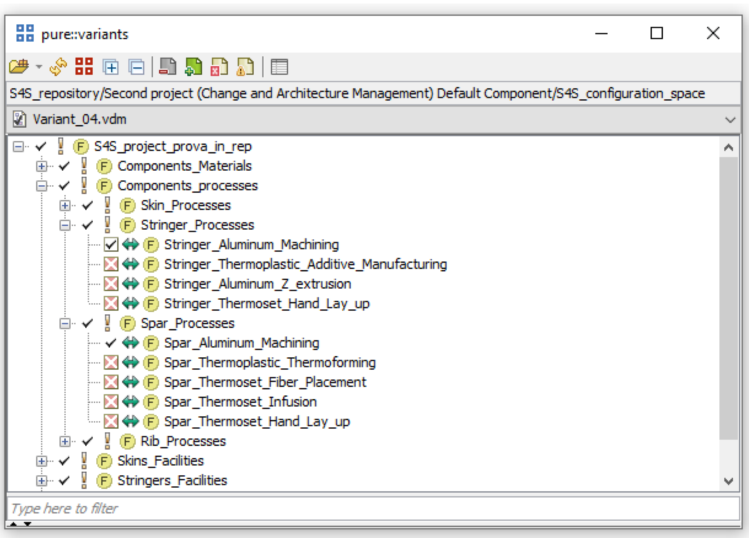
The selection of specific features enables the transformation of the 150% model into a 100% model, each selection is stored within a dedicated variant model, and multiple variant models can be generated. The matrix representation facilitates the identification of previously created models, increasing the ability to detect errors or duplicated variant models.



Model Elements	Level	Variant_...	Variant_...	Variant_...	Variant_...	Variant_...
prova_on_repo						
S4S_project_prova_in_rep		✓	✓	✓	✓	✓
Components_Materials	1	✓	✓	✓	✓	✓
Skin_Material	1.1	✓	✓	✓	✓	✓
Stringer_Material	1.2	✓	✓	✓	✓	✓
Spars_Material	1.3	✓	✓	✓	✓	✓
Spars_Aluminum	1.3.1	☑	☒	☒	☒	☒
Spars_Thermoset	1.3.2	☒	✓	☒	✓	✓
Spars_Thermoplastic	1.3.3	☒	☒	✓	☒	☒
Ribs_Material	1.4	✓	✓	✓	✓	✓
Components_processes	2	✓	✓	☑	✓	✓
Skin_Processes	2.1	✓	✓	✓	✓	✓
Stringer_Processes	2.2	✓	✓	✓	✓	✓
Spar_Processes	2.3	✓	✓	✓	✓	✓
Spar_Aluminum_Machining	2.3.1	✓	☒	☒	☒	☒
Spar_Thermoplastic_Thermofo...	2.3.2	☒	☒	☑	☒	☒
Spar_Thermoset_Fiber_Placem...	2.3.3	☒	☒	☒	☑	☒
Spar_Thermoset_Infusion	2.3.4	☒	☒	☒	☒	☒
Spar_Thermoset_Hand_Lay_up	2.3.5	☒	☑	☒	☒	☑
Rib_Processes	2.4	✓	✓	✓	✓	✓
Skins_Facilities	3	✓	✓	✓	✓	✓
Stringers_Facilities	4	✓	✓	✓	✓	✓
Spars_Facilities	5	✓	✓	✓	✓	✓
Ribs_Facilities	6	✓	✓	✓	✓	✓

Figure 29: Matrix overview of feature selection for different variants

The feature model is applied to the 150% model through an integration tool available in IBM Rhapsody, this tool has the ability to link system elements modelled in Rhapsody to specific features. As a result, the number of 100% models generated corresponds to the number of defined variant models.



pure::variants

S4S_repository/Second project (Change and Architecture Management) Default Component/S4S_configuration_space

Variant_04.vdm

```
✓ S4S_project_prova_in_rep
    ✓ Components_Materials
    ✓ Components_processes
        ✓ Skin_Processes
        ✓ Stringer_Processes
            ☑ Stringer_Aluminum_Machining
            ☒ Stringer_Thermoplastic_Additive_Manufacturing
            ☒ Stringer_Aluminum_Z_extrusion
            ☒ Stringer_Thermoset_Hand_Lay_up
        ✓ Spar_Processes
            ✓ Spar_Aluminum_Machining
            ☒ Spar_Thermoplastic_Thermoforming
            ☒ Spar_Thermoset_Fiber_Placement
            ☒ Spar_Thermoset_Infusion
            ☒ Spar_Thermoset_Hand_Lay_up
        ✓ Rib_Processes
    ✓ Skins_Facilities
    ✓ Stringers_Facilities
```

Type here to filter

Figure 30: Pure::Variants integration tool present in IBM Rhapsody

4.3 Tools supporting evaluation and decision making

To implement the methodology described in Section 3.6 a series of tools are used. Python-based codes implement the equation for the value-model theory and calculate the performance of the production domain and the product domain.

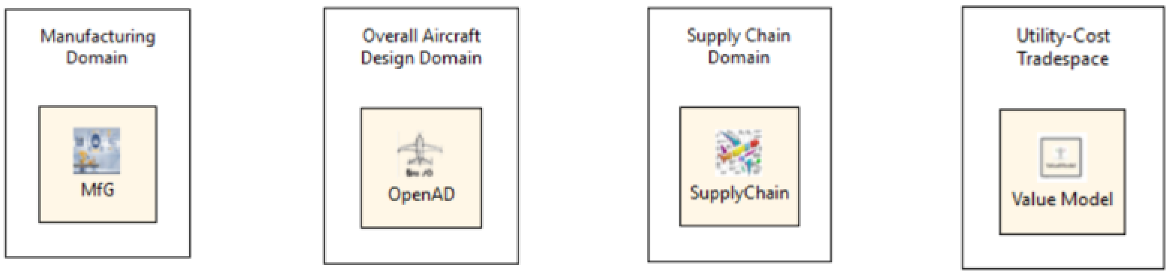


Figure 31: The four codes implemented by the DLR to calculate the value of a S4S architecture

The four codes presented in Figure 31 have been already implemented by the DLR Institute of System Architectures in Aeronautics. If the reader desires a better understanding of the tools Donelli Giuseppa's Thesis can provide more inside. In this work a small introduction is presented.

Python codes are connected to exchange information to generate the value-driven trade space and achieve the automatic execution of these methodologies. CPACS (Common Parametric Aircraft Configuration Schema), developed in Hamburg by the German Aerospace Center, is the common language that enables communication between the domains. It is an XML file that saves the most important properties of the aircraft, but it can be expanded using toll-specific branches. The Python code can extract input from CPACS and upload outputs in the same file and since all the CPACS communicate in the same language they can be connected in a workflow.

The workflow is implemented in RCE (Remote Component Environment)(DLR, 2020), who can connect the different domains and recreate the workflow needed to execute the analysis.

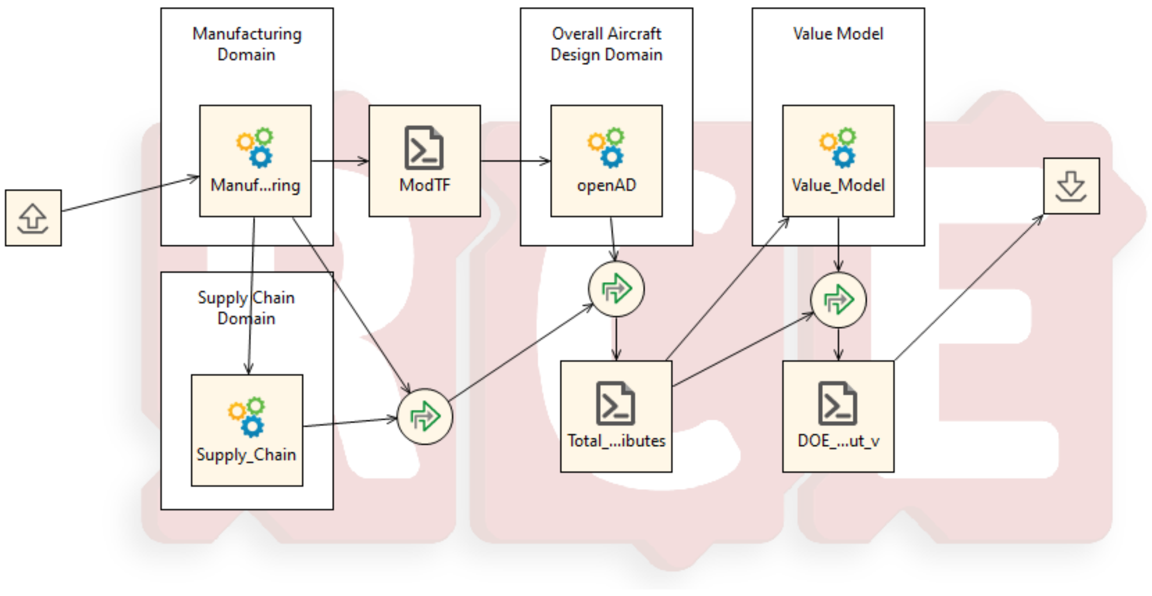
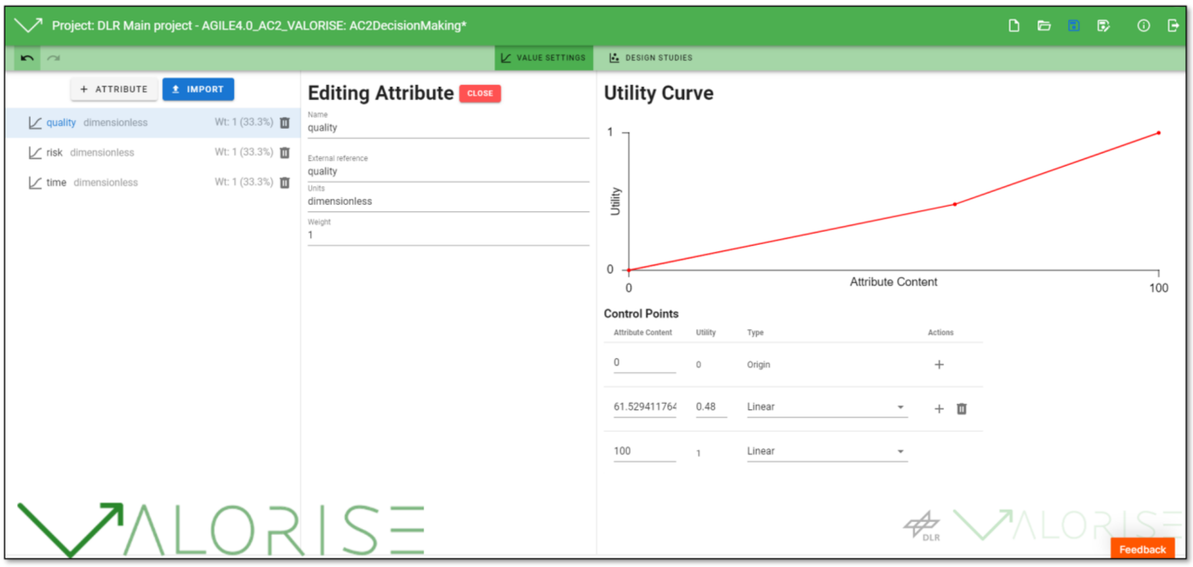


Figure 32: An example of connection of tools in RCE

VALORISE, an interactive dashboard developed by DLR, stands for 'Value-driven trade space visualization, exploitation and assessment'. It supports the modelling of value-driven decision-making processes, facilitates analysis of real-time strategic scenarios and explores value-driven trade spaces to identify robust solutions. VALORISE is an interactive dashboard that decision-makers can use directly to identify the best solution. It can also be used as an executable tool integrated into a tool chain to explore many weight combinations and/or utility functions.

Decision-makers are able to interactively draw utility functions in order to represent their expectations with respect to each selected attribute. They can also set several weight combinations in order to analyze the scenario of interest. Many scenarios can then be investigated in real-time in VALORISE.



quality	dimensionless	Wt: 1 (33.3%)
risk	dimensionless	Wt: 1 (33.3%)
time	dimensionless	Wt: 1 (33.3%)

Editing Attribute

Name: quality

External reference: quality

Units: dimensionless

Weight: 1

Utility Curve

Control Points

Attribute Content	Utility	Type	Actions
0	0	Origin	+
61.529411764	0.48	Linear	+
100	1	Linear	

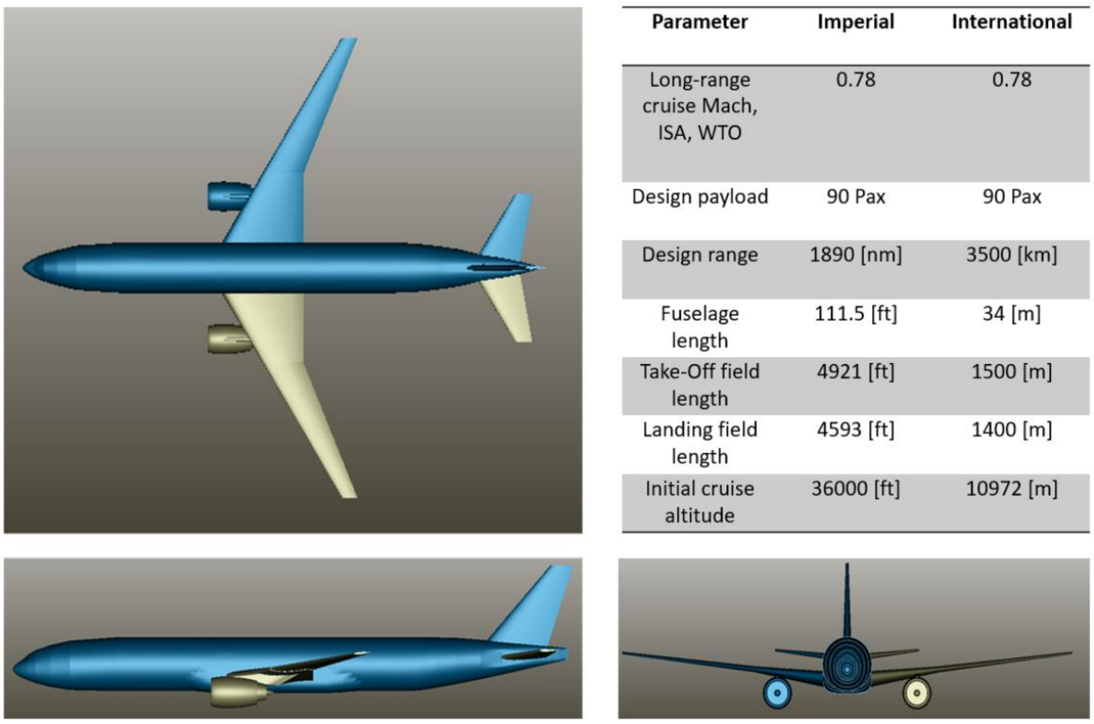
Figure 33: VALORISE Dashboard settings

5 Methodology Application

The methodology proposed in Section 3 and the tools described in the previous section are applied to an aeronautical case study to demonstrate the advantages of using such a methodology. In this section, first, the assumptions of this case study are provided and then the results are discussed.

5.1 Introduction to the application test case

The case study refers to the design, manufacturing, and supply chain of a horizontal tail plane (HTP) of a 90-passenger regional aircraft, whose main characteristics of the aircraft are reported in Figure 34 and the main parameters are reported in Table 4.



Parameter	Imperial	International
Long-range cruise Mach, ISA, WTO	0.78	0.78
Design payload	90 Pax	90 Pax
Design range	1890 [nm]	3500 [km]
Fuselage length	111.5 [ft]	34 [m]
Take-Off field length	4921 [ft]	1500 [m]
Landing field length	4593 [ft]	1400 [m]
Initial cruise altitude	36000 [ft]	10972 [m]

Figure 34:Three views and main characteristics of the reference aircraft(Donelli Giuseppa, 2024)

GEOMETRICAL PARAMETERS			MASS BREAKDOWN		
PARAMETER	Unit	Value	PARAMETER	Unit	Value
S	m2	81.4	MTOW	kg	44898
b	m	27.186	MZFW	kg	37309
AR	-	9.09	MOEW	kg	28129
MAC	m	3.796	MEW	kg	24874
lf	m	34	MPAYLOAD	kg	9180

Table 4:Reference Aircraft Main Parameters

The HTP under consideration is composed of four kind of structural components, skin panels, stringers, ribs and spars. The whole architecture is made up of 2 panels, 30 stringers, 2 spars and 20 ribs.

The S4S framework involves the supply chain and the manufacturing system as enabling systems and HTP as product or system of interest. The variability of the system is introduced considering different materials, processes and facilities for the manufacturing of the system. In Table 5 are reported the exact numbers.

	SKINS	STRINGERS	SPARS	RIBS
N° MATERIALS	2	3	3	4
N° MANUFACTURING PROCESS	6	4	5	5
N° ENTERPRISES	20	20	20	21

Table 5: Numbers of materials, processes, and enterprises for each component.

The possible materials for each component are illustrated in Table 6.

	MATERIALS
Skins	Aluminum; Thermoset
Stringers	Aluminum; Thermoset; Thermoplastic
Spars	Aluminum; Thermoset; Thermoplastic
Ribs	Aluminum; Thermoset; Thermoplastic; Titanium

Table 6: Possible materials choice for each component.

The possible processes for each component are, instead, illustrated in Table 7.

	PROCESS
Skins	Aluminum Machining; Aluminum Stretch Formed and press Formed; Thermoset Hand Lay up; Thermoset Automated Tape Lay up; Thermoset Infusion Processes; Thermoset Fiber Placement
Stringers	Aluminum Machining; Aluminum Z-extrusion; Thermoset Hand Lay up; Thermoplastic Additive Manufacturing
Spars	Aluminum Machining; Thermoset Hand Lay up; Thermoset Infusion Processes; Thermoset Fiber Placement; Thermoplastic Thermoforming
Ribs	Aluminum Machining; Aluminum Z-extrusion; Thermoset Hand Lay up; Thermoplastic Thermoforming; Titanium machining

Table 7: Possible process choice for each component.

The facilities are divided into three categories. Five are OEM sites, seven of them are Suppliers Tier 1 and nine are Suppliers Tier 2. For each of them, position, competences and processes available are known.

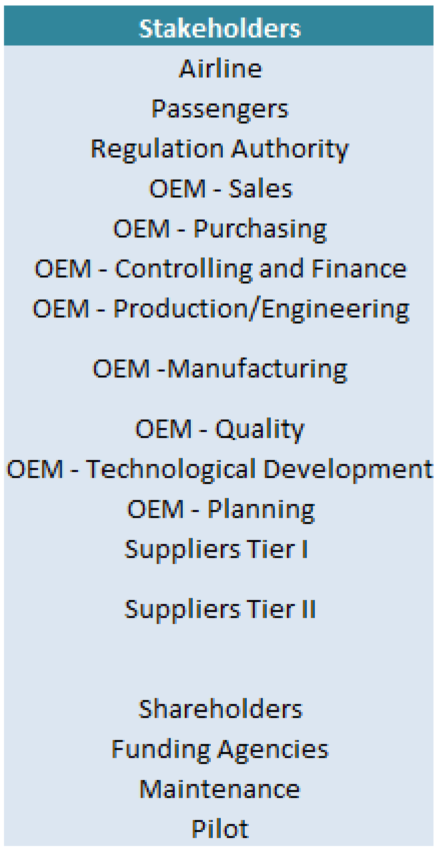
The supply chain is responsible for combining all the components to deliver the final product. In the case under consideration, three main assembly phases are considered. The first is to join stringers and skin panels. The second is the main assembly line where all the components are joined together, and the final product is the HTP. The last phase is the assembly of the HTP to the aircraft. Different processes can be used to perform the assembly and different facilities. However, in this test case, the OEM site 1 is the only facility that will perform the assembly, the assembly processes are fixed and will not change.

Following the methodology presented in Section 3, first the requirements are presented, following the base architecture is design and the variability is introduced to the system, finally the best solution is found as a result of the MAUT analysis.

5.2 Requirement

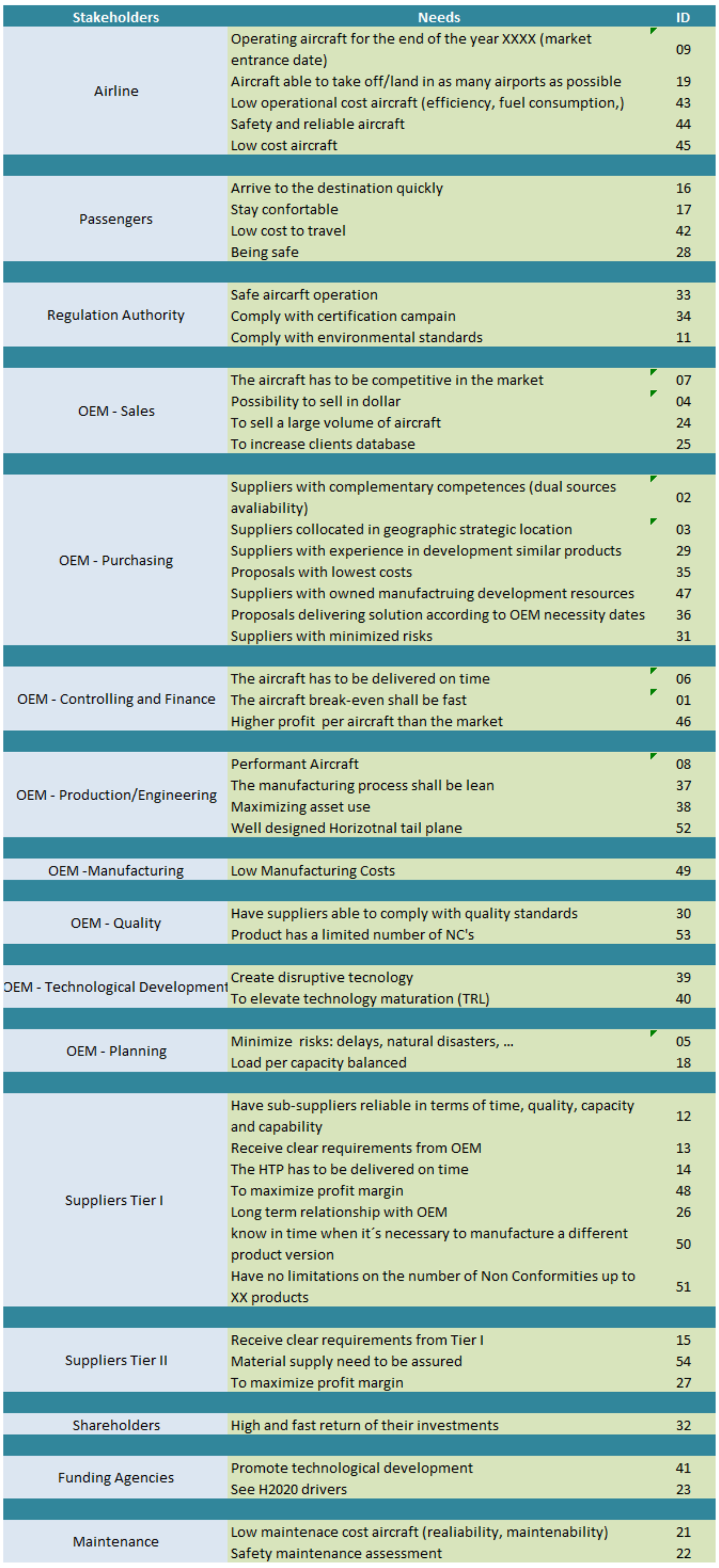
In Section 3.3 is introduced the methodology to translate stakeholders' needs into stakeholders' requirements and then derives system requirements. In this section, the methodology is applied to the test case under consideration. First needs and stakeholders are presented, then the requirements are derived, and finally, requirements expression is imported into IBM Rhapsody to be connected to system components and behaviors.

Stakeholders have been identified through a combination of interviews and literature research. Figure 35 groups the stakeholders involved and shows them in a hierarchy. They are part of the production of the system, or they have an interest in the product, also, since the HTP is part of the aircraft stakeholders linked to the whole aircraft are identified, such as passengers or airlines. For the OEM, different sections of the company are shown, each with different interests. In addition, Figure 36 lists and identifies the needs discussed, using an identification code (ID).



Stakeholders
Airline
Passengers
Regulation Authority
OEM - Sales
OEM - Purchasing
OEM - Controlling and Finance
OEM - Production/Engineering
OEM -Manufacturing
OEM - Quality
OEM - Technological Development
OEM - Planning
Suppliers Tier I
Suppliers Tier II
Shareholders
Funding Agencies
Maintenance
Pilot

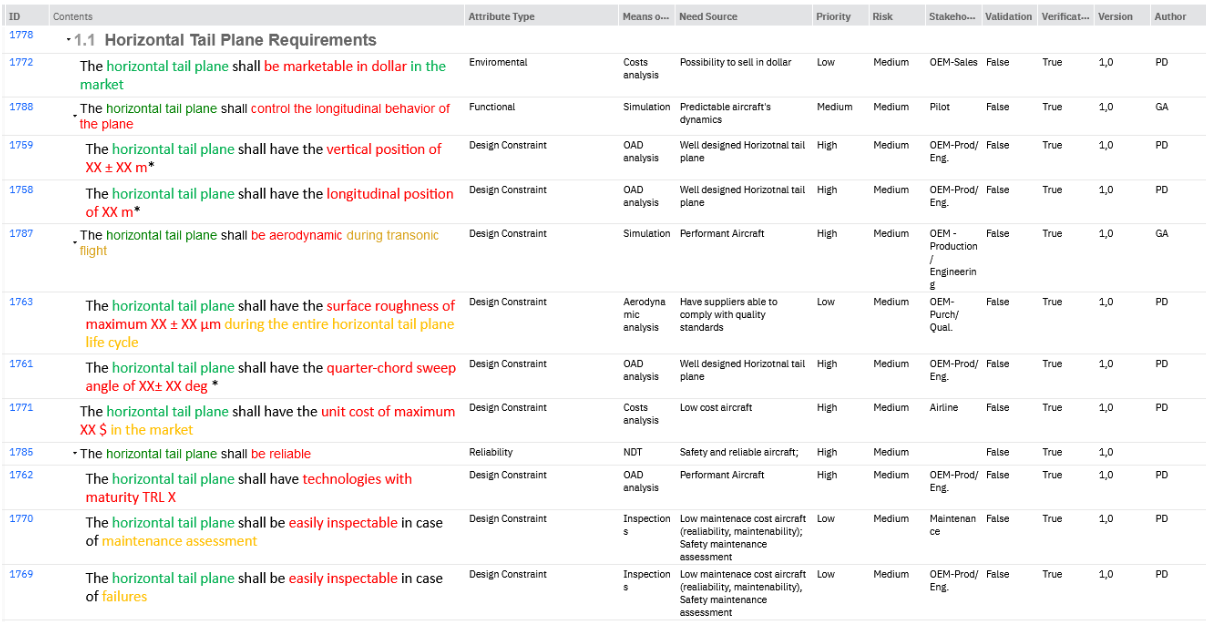
Figure 35: Stakeholders List



Stakeholders	Needs	ID
Airline	Operating aircraft for the end of the year XXXX (market entrance date)	09
	Aircraft able to take off/land in as many airports as possible	19
	Low operational cost aircraft (efficiency, fuel consumption,)	43
	Safety and reliable aircraft	44
	Low cost aircraft	45
Passengers	Arrive to the destination quickly	16
	Stay confortable	17
	Low cost to travel	42
	Being safe	28
Regulation Authority	Safe aircarft operation	33
	Comply with certification campain	34
	Comply with environmental standards	11
OEM - Sales	The aircraft has to be competitive in the market	07
	Possibility to sell in dollar	04
	To sell a large volume of aircraft	24
	To increase clients database	25
OEM - Purchasing	Suppliers with complementary competences (dual sources avaliability)	02
	Suppliers collocated in geographic strategic location	03
	Suppliers with experience in development similar products	29
	Proposals with lowest costs	35
	Suppliers with owned manufactruing development resources	47
	Proposals delivering solution according to OEM necessity dates	36
	Suppliers with minimized risks	31
OEM - Controlling and Finance	The aircraft has to be delivered on time	06
	The aircraft break-even shall be fast	01
	Higher profit per aircraft than the market	46
OEM - Production/Engineering	Performant Aircraft	08
	The manufacturing process shall be lean	37
	Maximizing asset use	38
	Well designed Horizotnal tail plane	52
OEM -Manufacturing	Low Manufacturing Costs	49
OEM - Quality	Have suppliers able to comply with quality standards	30
	Product has a limited number of NC's	53
OEM - Technological Development	Create disruptive tecnology	39
	To elevate technology maturation (TRL)	40
OEM - Planning	Minimize risks: delays, natural disasters, …	05
	Load per capacity balanced	18
Suppliers Tier I	Have sub-suppliers reliable in terms of time, quality, capacity and capability	12
	Receive clear requirements from OEM	13
	The HTP has to be delivered on time	14
	To maximize profit margin	48
	Long term relationship with OEM	26
	know in time when it´s necessary to manufacture a different product version	50
	Have no limitations on the number of Non Conformities up to XX products	51
Suppliers Tier II	Receive clear requirements from Tier I	15
	Material supply need to be assured	54
	To maximize profit margin	27
Shareholders	High and fast return of their investments	32
Funding Agencies	Promote technological development	41
	See H2020 drivers	23
Maintenance	Low maintenace cost aircraft (realiability, maintenability)	21
	Safety maintenance assessment	22

Figure 36: Stakeholders Needs and their Identification number

The identified needs are then translated using the methodology previously presented. Within IBM Doors, requirements are derived from needs following the rules and patterns introduced in the methodology. In this software, it was decided to use a screen displaying all the predefined type attributes. In column 5 from left, the needs from which each requirement originates are displayed. If a requirement is not directly referable to a need but rather to a parent requirement this relationship is shown in the same box as the needs.

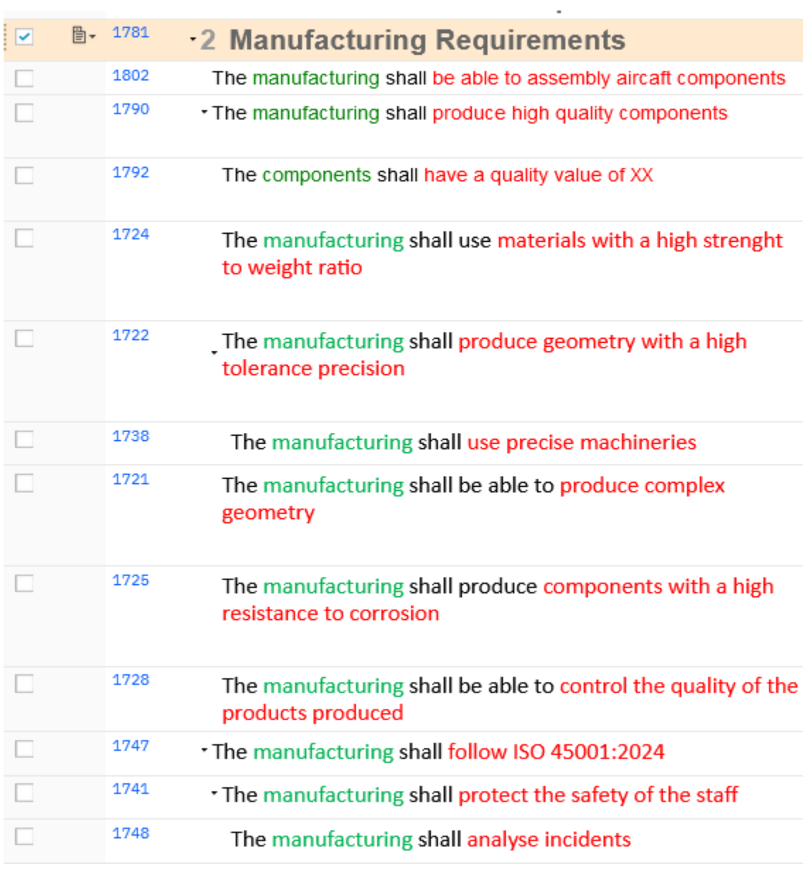


ID	Contents	Attribute Type	Means o...	Need Source	Priority	Risk	Stakeho...	Validation	Verificat...	Version	Author
1778	1.1 Horizontal Tail Plane Requirements										
1772	The horizontal tail plane shall be marketable in dollar in the market	Enviromental	Costs analysis	Possibility to sell in dollar	Low	Medium	OEM-Sales	False	True	1,0	PD
1788	The horizontal tail plane shall control the longitudinal behavior of the plane	Functional	Simulation	Predictable aircraft's dynamics	Medium	Medium	Pilot	False	True	1,0	GA
1759	The horizontal tail plane shall have the vertical position of XX ± XX m*	Design Constraint	OAD analysis	Well designed Horizotnal tail plane	High	Medium	OEM-Prod/ Eng.	False	True	1,0	PD
1758	The horizontal tail plane shall have the longitudinal position of XX m*	Design Constraint	OAD analysis	Well designed Horizotnal tail plane	High	Medium	OEM-Prod/ Eng.	False	True	1,0	PD
1787	The horizontal tail plane shall be aerodynamic during transonic flight	Design Constraint	Simulation	Performant Aircraft	High	Medium	OEM - Production / Engineering	False	True	1,0	GA
1763	The horizontal tail plane shall have the surface roughness of maximum XX ± XX µm during the entire horizontal tail plane life cycle	Design Constraint	Aerodynamic analysis	Have suppliers able to comply with quality standards	Low	Medium	OEM-Purch/ Qual.	False	True	1,0	PD
1761	The horizontal tail plane shall have the quarter-chord sweep angle of XX± XX deg *	Design Constraint	OAD analysis	Well designed Horizotnal tail plane	High	Medium	OEM-Prod/ Eng.	False	True	1,0	PD
1771	The horizontal tail plane shall have the unit cost of maximum XX $ in the market	Design Constraint	Costs analysis	Low cost aircraft	High	Medium	Airline	False	True	1,0	PD
1785	The horizontal tail plane shall be reliable	Reliability	NDT	Safety and reliable aircraft;	High	Medium		False	True	1,0	
1762	The horizontal tail plane shall have technologies with maturity TRL X	Design Constraint	OAD analysis	Performant Aircraft	High	Medium	OEM-Prod/ Eng.	False	True	1,0	PD
1770	The horizontal tail plane shall be easily inspectable in case of maintenance assessment	Design Constraint	Inspections	Low maintenace cost aircraft (realiability, maintenability); Safety maintenance assessment	Low	Medium	Maintenance	False	True	1,0	PD
1769	The horizontal tail plane shall be easily inspectable in case of failures	Design Constraint	Inspections	Low maintenace cost aircraft (realiability, maintenability), Safety maintenance assessment	Low	Medium	OEM-Prod/ Eng.	False	True	1,0	PD

Figure 37: Requirement view inside IBM DOORS with attributes

The text of the requirement is highlighted in different colors to echo the requirement patterns presented in Table 2. Green represents the system to which the requirement refers, red is the function, yellow is the condition, and brown is the performance.

The visualization of the requirements hierarchy is possible through the visualization in Figure 38, in the image requirement 1790 is the parent of requirements 1792 and 1724, this is displayed by the arrow on the side of the requirement. A better visualization is possible through the tree view that can be called up within the software, also this option enables the possibility of displaying the links and relationships between requirements, Figure 26 shows this mode.



✓	1781	2 Manufacturing Requirements
☐	1802	The manufacturing shall be able to assembly aircaft components
☐	1790	The manufacturing shall produce high quality components
☐	1792	The components shall have a quality value of XX
☐	1724	The manufacturing shall use materials with a high strenght to weight ratio
☐	1722	The manufacturing shall produce geometry with a high tolerance precision
☐	1738	The manufacturing shall use precise machineries
☐	1721	The manufacturing shall be able to produce complex geometry
☐	1725	The manufacturing shall produce components with a high resistance to corrosion
☐	1728	The manufacturing shall be able to control the quality of the products produced
☐	1747	The manufacturing shall follow ISO 45001:2024
☐	1741	The manufacturing shall protect the safety of the staff
☐	1748	The manufacturing shall analyse incidents

Figure 38: Requirement view inside IBM DOORS

For the supply chain the requirements sought concern OEMs and suppliers, for the manufacturing system the requirements involve the system itself and the machinery that makes it up. A further level of detail is not necessary for the purposes of this exercise, but for a real system the systems must be further specified and consequently the requirements reach a higher level of detail.

Once the requirements formulation is finished, the system architecture can begin. In this specific case, to start the next phase, it was decided to import the requirements into IBM Rhapsody and to model the relationships between requirements and system components within this software.

5.3 Behavior and Composition

Once the requirements have been translated from the needs and linked to other requirements derived it is possible to start construct the architecture of the system. In this section the methodology introduced in Section 3.4 is exploited for the system under consideration.

The first step is to delimit the system using the Use case Diagram to identify the context, actors, and use case. The diagram is shown in Figure 39.

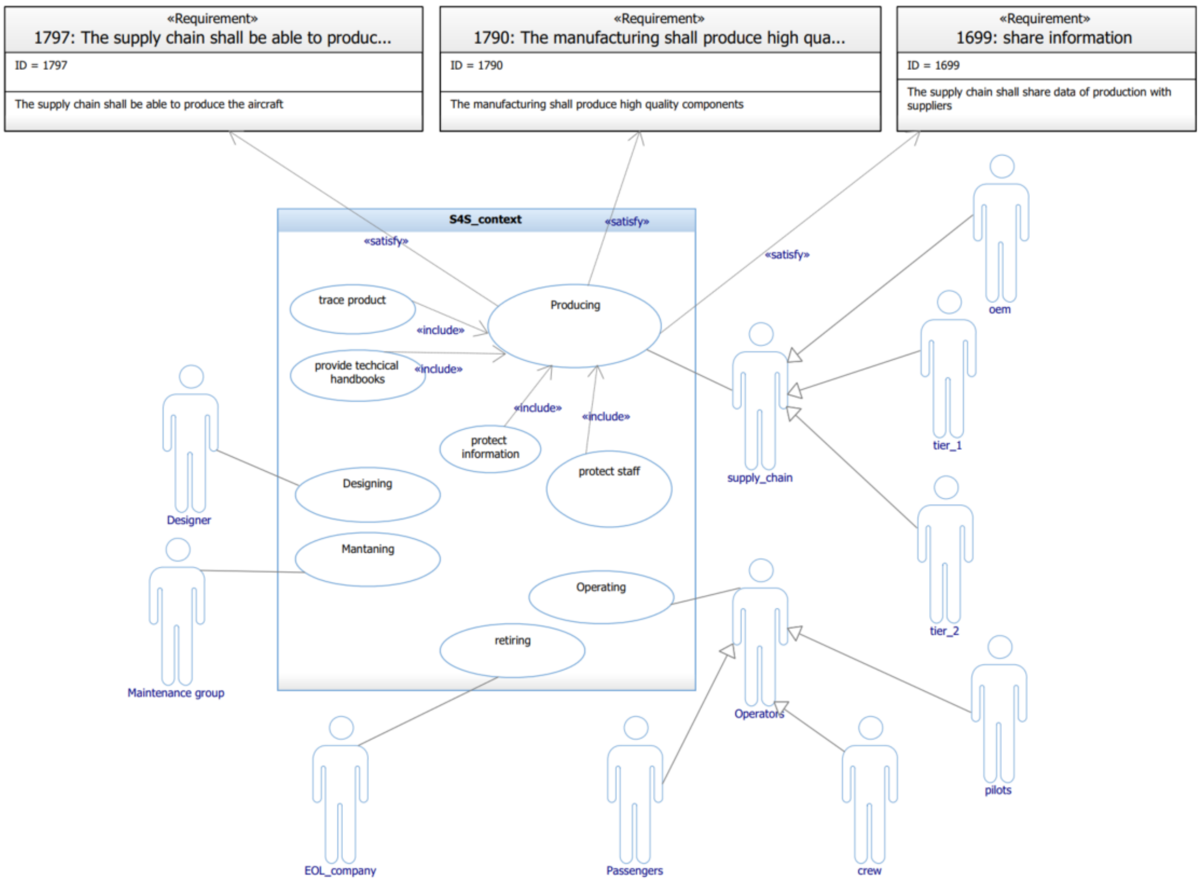


Figure 39: Use Case Diagram for the S4S context

The actors considered are part of the staff, customers, or groups of people who interact with the system in one or more moments of the system life cycle. That is the reason why the operators, companies responsible for the disposal of the system at the end of its life (EOL_company), and the maintenance staff are considered. However, the focus is on the supply chain, divided into OEM and Suppliers Tier 1 and 2, who have the function to produce the aircraft. Three requirements are considered to show how from the Use Case, activities, composition of the system, and connection between components can be achieved. The functional requirement 1699 "*the supply chain shall share data of production with suppliers*" is satisfied by the use of a sequence diagram that shows how messages between elements of the supply chain allow to sharing of production information. The functional requirement 1797 "*the supply chain shall be able to produce the aircraft*" is used to demonstrate that from a requirement it is possible to derive information about the structure of the system. Finally, requirement 1790 "*the manufacturing shall produce high quality components*" is used to demonstrate how it is possible to have information about specific components of the system. Figure 40 summarizes these three processes in 3 flows that start from the *producing* use case.

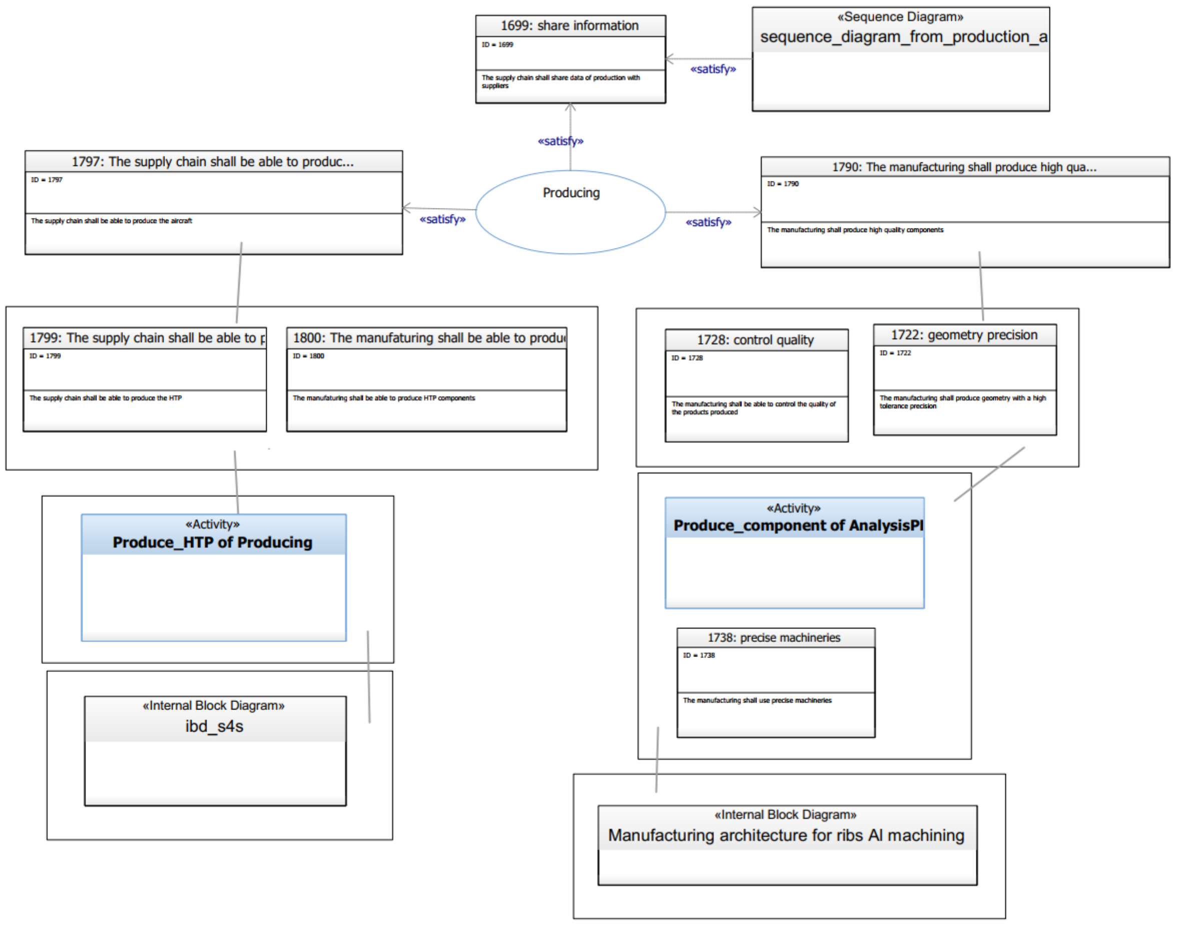

«Sequence Diagram»
sequence_diagram_from_production_a
1699: share information
ID = 1699
The supply chain shall share data of production with suppliers
«satisfy»
«satisfy»
1797: The supply chain shall be able to produc...
ID = 1797
The supply chain shall be able to produce the aircraft
«satisfy»
Producing
«satisfy»
1790: The manufacturing shall produce high qua...
ID = 1790
The manufacturing shall produce high quality components
1799: The supply chain shall be able to p
ID = 1799
The supply chain shall be able to produce the HTP
1800: The manufaturing shall be able to produ
ID = 1800
The manufaturing shall be able to produce HTP components
1728: control quality
ID = 1728
The manufacturing shall be able to control the quality of the products produced
1722: geometry precision
ID = 1722
The manufacturing shall produce geometry with a high tolerance precision
«Activity»
Produce_HTP of Producing
«Activity»
Produce_component of AnalysisPl
1738: precise machineries
ID = 1738
The manufacturing shall use precise machineries
«Internal Block Diagram»
ibd_s4s
«Internal Block Diagram»
Manufacturing architecture for ribs Al machining

Figure 40: Summary of the three processes used to derive the architecture

5.3.1 Sequence Diagram

As introduced in Section 3.4, functional requirements are linked to the use case to assign a function that the system must perform to satisfy stakeholders' expectations. In this case the supply chain, who is the actor in the production use case, has to be able to interact with and share information about production with suppliers, transport system and OEM. The sequence diagram represents the message that the system components share to update the information in their possession.

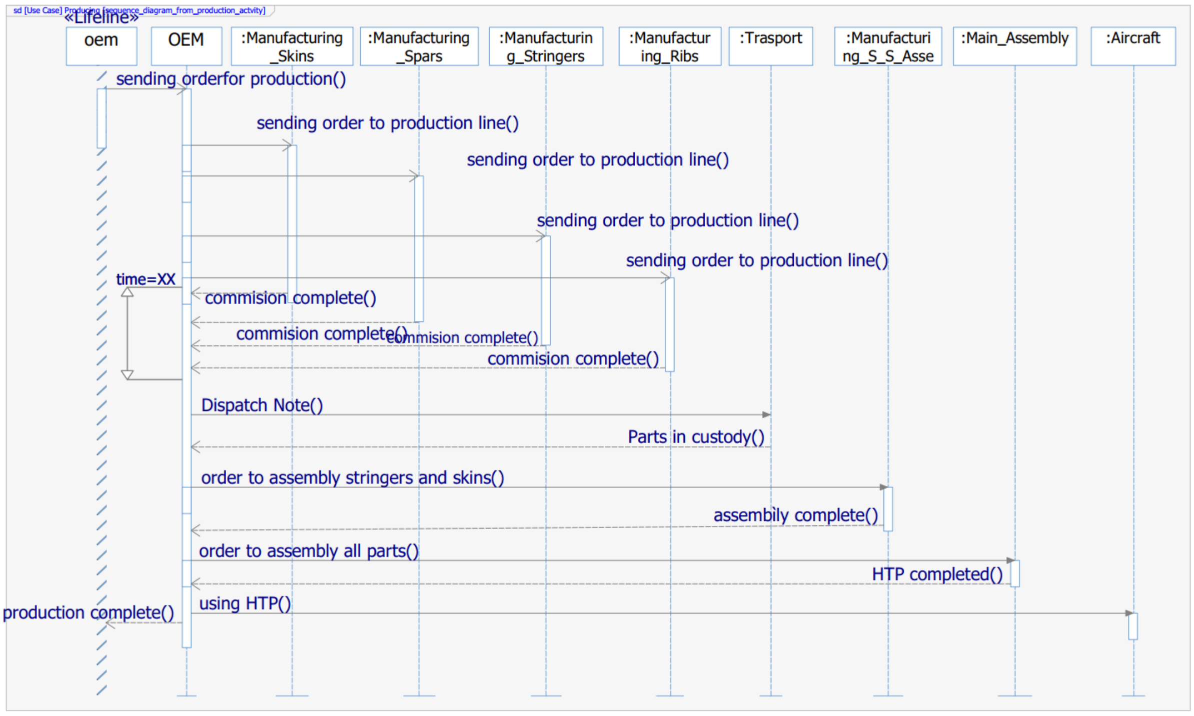


Figure 41: Sequence Diagram showing the messages exchanged between supply chain, manufacturing, and actors

5.3.2 System structure

The left part of the diagram in Figure 40 is, instead, used to show the procedure to structure the system. The derivation of requirement 1797 into requirement 1799 and 1800 has been done in the requirement phase. From these two requirements it is possible to understand the two functions that the supply chain has to perform. It is possible to build an activity that shows the actions that the systems must implement to satisfy the requirements. Figure 42 shows the diagram.

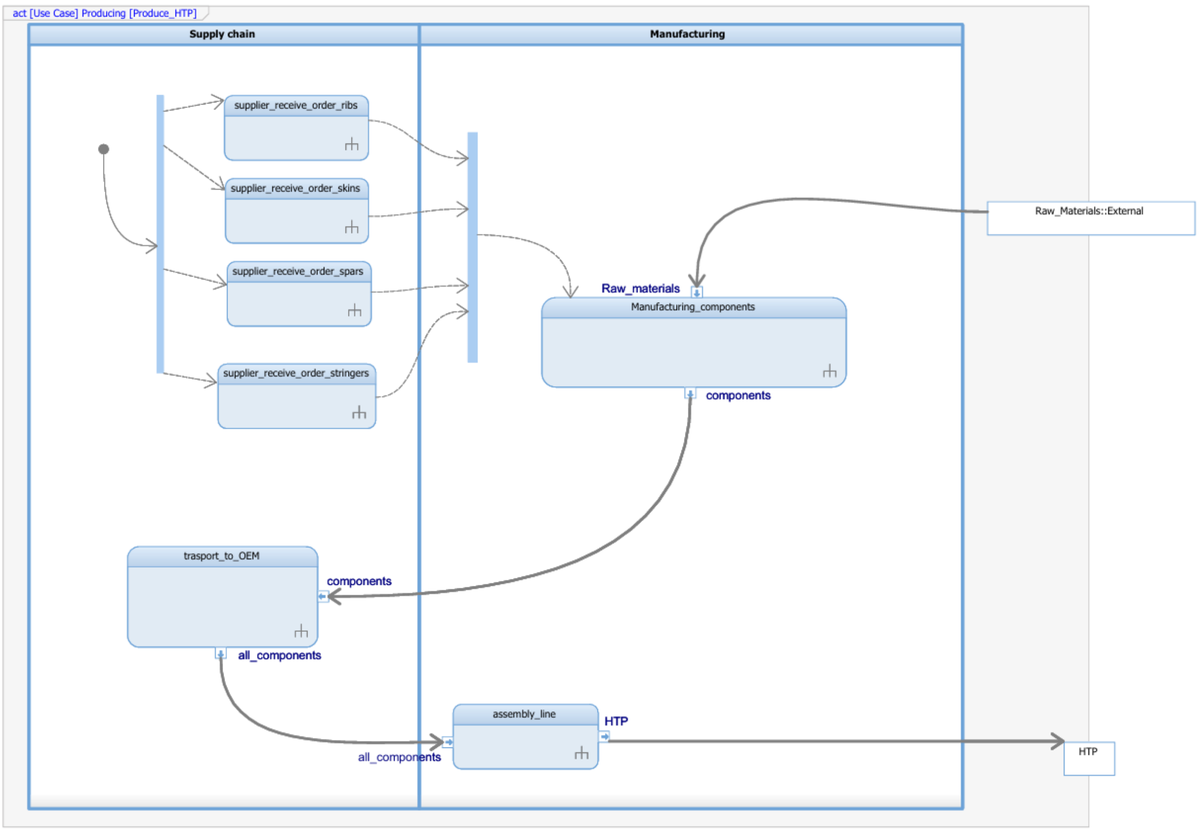


Figure 42: Activity Diagram that specifies the *producing* use case

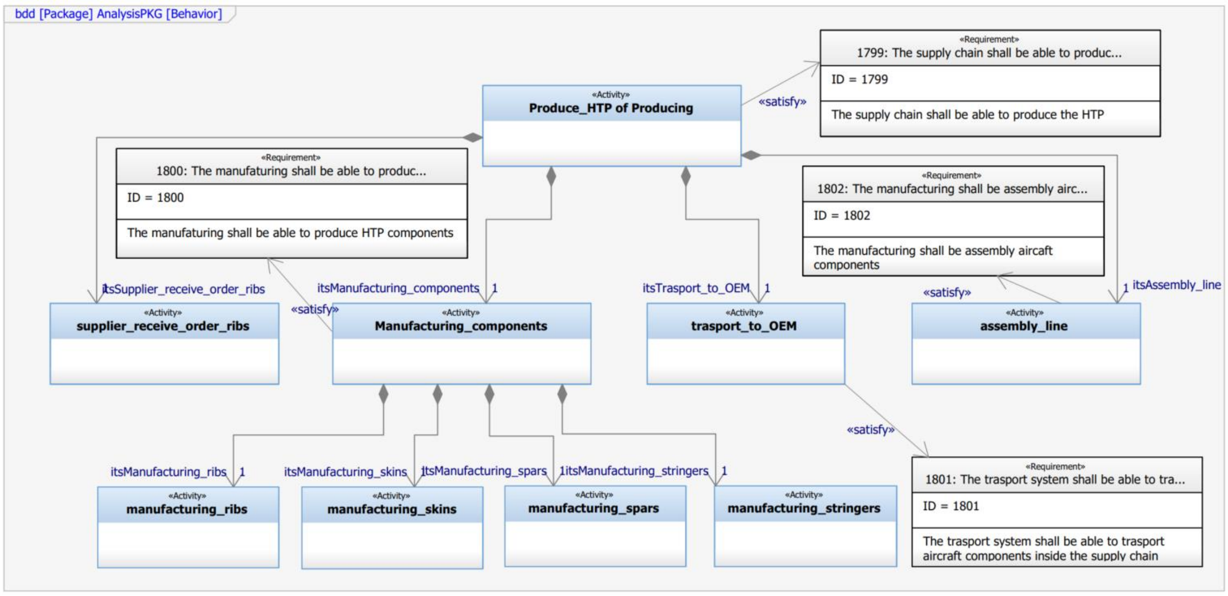

bdd [Package] AnalysisPKG [Behavior]
«Activity» Produce_HTP of Producing
«satisfy»
«Requirement» 1799: The supply chain shall be able to produc...
ID = 1799
The supply chain shall be able to produce the HTP
«Requirement» 1800: The manufaturing shall be able to produc...
ID = 1800
The manufaturing shall be able to produce HTP components
«Requirement» 1802: The manufacturing shall be assembly airc...
ID = 1802
The manufacturing shall be assembly aircraft components
itsSupplier_receive_order_ribs
itsManufacturing_components 1
«satisfy»
itsTrasport_to_OEM 1
«satisfy»
1 itsAssembly_line
«Activity» supplier_receive_order_ribs
«Activity» Manufacturing_components
«Activity» trasport_to_OEM
«Activity» assembly_line
«satisfy»
itsManufacturing_ribs 1
itsManufacturing_skins 1
itsManufacturing_spars 1
itsManufacturing_stringers 1
«Activity» manufacturing_ribs
«Activity» manufacturing_skins
«Activity» manufacturing_spars
«Activity» manufacturing_stringers
«Requirement» 1801: The trasport system shall be able to tra...
ID = 1801
The trasport system shall be able to trasport aircraft components inside the supply chain

Figure 43: BDD of all the activities inside the *producing* use case linked to the requirements from which they derive

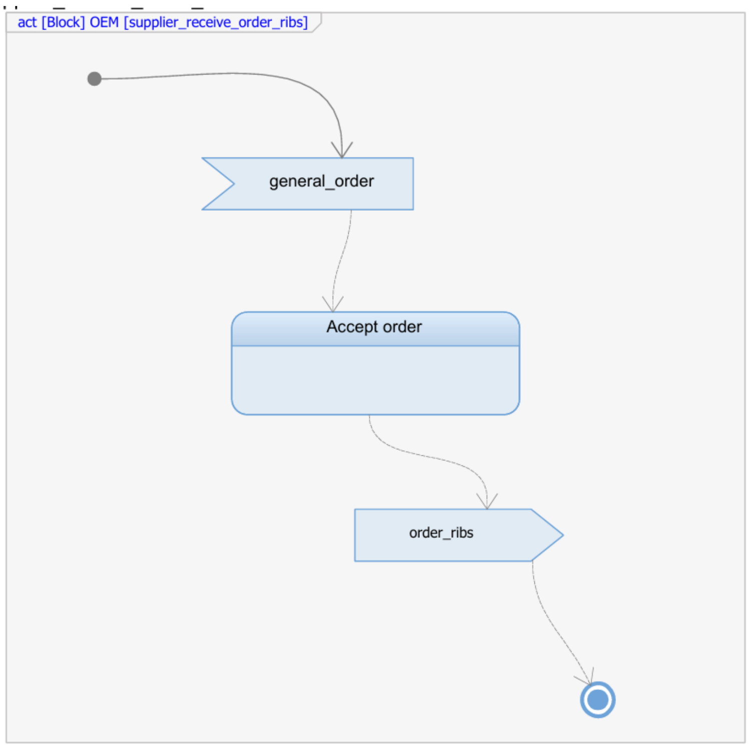
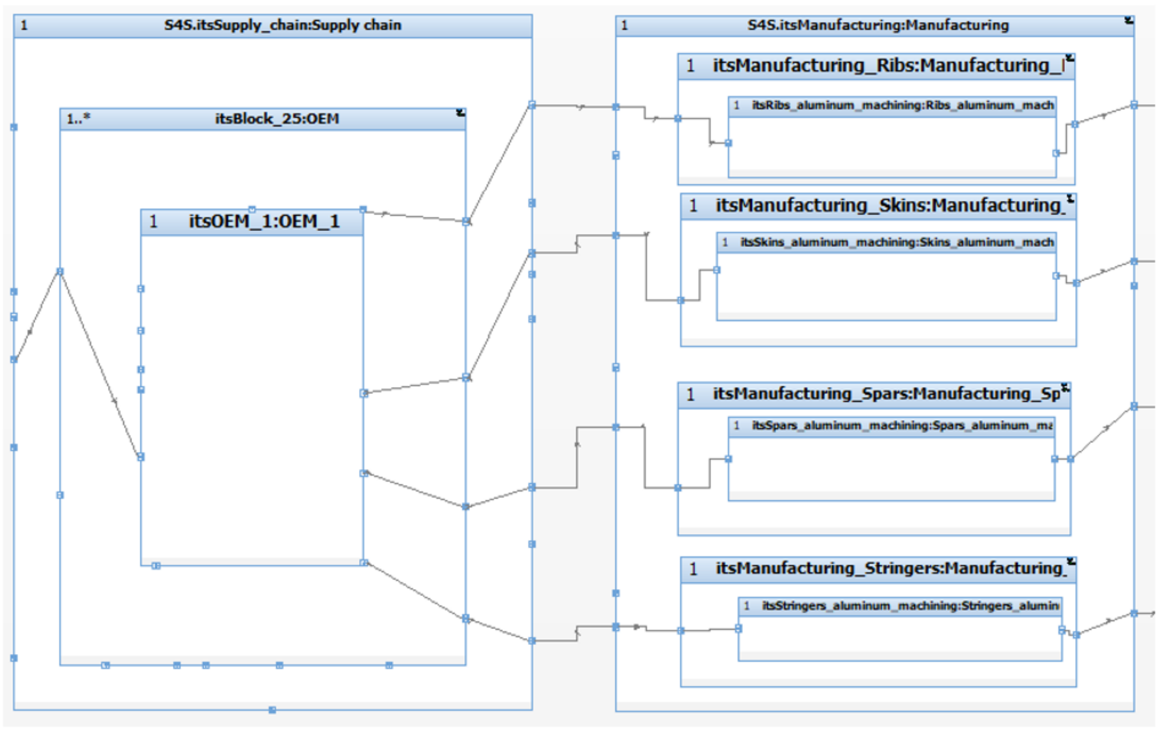
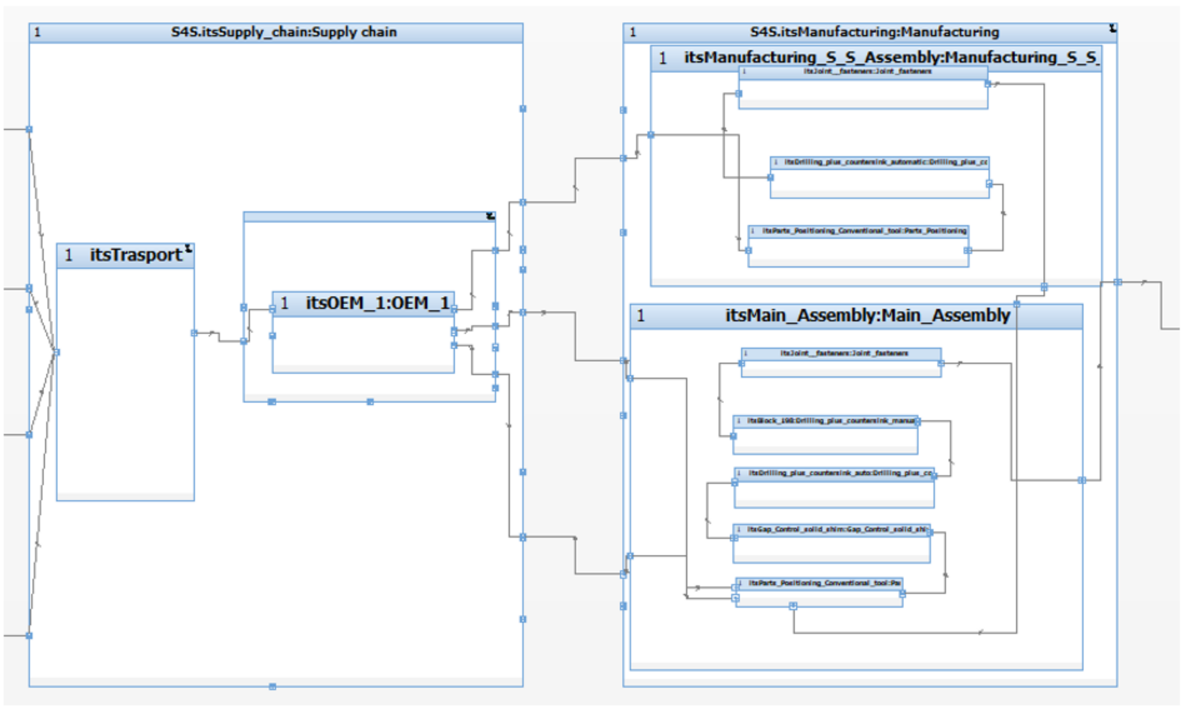


Figure 44: Example of *callBehavior*

The swimlanes are used to represent who is responsible for implementing specific actions. In this case, the supply chain is responsible for implementing the actions in the left box and the manufacturing system the actions on the left. The fork inside the actions represents a particular action called "*callBehavior*", which represent an action that is used more than once. Every "*callBehavior*" is represented by another activity diagram, an example is given in Figure 44 where the "*supplier receiving order*" is displayed. The whole activity diagram is able to represent the actions to build the HTP (last action needed is the assembly of the components) while the ability to produce the specific components are assigned to the manufacturing. From this activity and considering the actions inside the "*callBehavior*" is possible to identify how the system is structured. The BDD is shown but the Internal Block Diagram in Figure 45 is actually more interesting since it is possible to recognize the connection between action and component.


1 S4S.itsSupply_chain:Supply chain
1..* itsBlock_25:OEM
1 itsOEM_1:OEM_1
1 S4S.itsManufacturing:Manufacturing
1 itsManufacturing_Ribs:Manufacturing_I
1 itsRibs_aluminum_machining:Ribs_aluminum_mach
1 itsManufacturing_Skins:Manufacturing_
1 itsSkins_aluminum_machining:Skins_aluminum_mach
1 itsManufacturing_Spars:Manufacturing_Sp
1 itsSpars_aluminum_machining:Spars_aluminum_ma
1 itsManufacturing_Stringers:Manufacturing_
1 itsStringers_aluminum_machining:Stringers_alumin


1 S4S.itsSupply_chain:Supply chain
1 itsTrasport
1 itsOEM_1:OEM_1
1 S4S.itsManufacturing:Manufacturing
1 itsManufacturing_S_S_Assembly:Manufacturing_S_S_
1 itsJoint__fasteners:Joint_fasteners
1 itsDrilling_plus_countersink_automatic:Drilling_plus_cc
1 itsParts_Positioning_Conventional_tool:Parts_Positioning
1 itsMain_Assembly:Main_Assembly
1 itsJoint__fasteners:Joint_fasteners
1 itsBlock_198:Drilling_plus_countersink_manua
1 itsDrilling_plus_countersink_auto:Drilling_plus_cc
1 itsGap_Control_solid_shim:Gap_Control_solid_shi
1 itsParts_Positioning_Conventional_tool:Pa

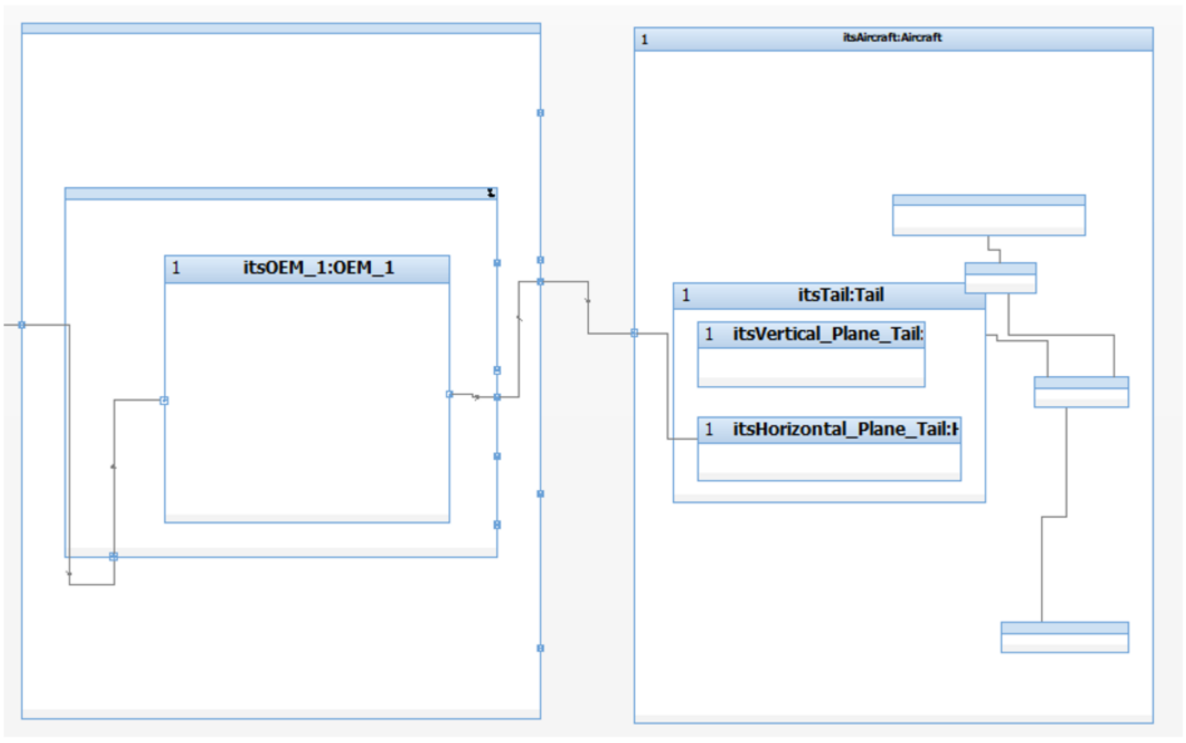
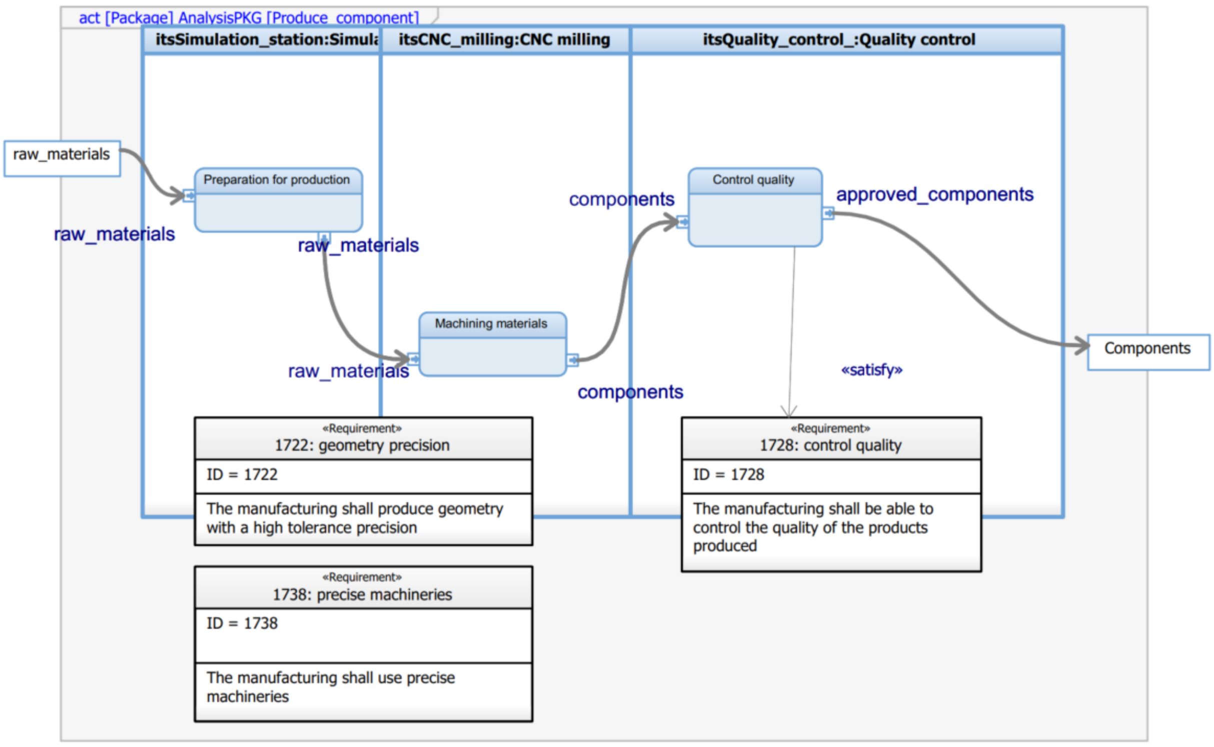


Figure 45: IBD of all S4S

The diagram is to be read from left to right, the first block represents the components of the supply chain who receive the order to manufacture the component, this block is linked to the action "*supplier receive order*" in the activity diagram. The second block represents the beginning of production by the manufacturing system. Only the process used is shown in this diagram, the components of the process are displayed in the internal block diagram of the process. The manufacturing system is responsible for acting "*manufacturing X*". Then the transportation sub-system is responsible for executing the action "*Trasport to OEM*" and finally the OEM, who is part of the supply chain system, is responsible for the assembly.

5.3.3 System components

The right arm of the diagram in Figure 40 is used to demonstrate how a requirement can lead to the selection of a specific component. The two requirements 1728 and 1722 are used to link requirement 1790 to the rest of the structure. These two requirements are linked to two actions inside the *callBehavior "produce component",* which is part of the "*Produce HTP*" presented in Figure 42. From these requirements and other performance requirements it is possible to select two components for the Aluminum Machining process, the CNC machine, which can achieve the performance required, and the Quality control machine which can perform the quality action.



act [Package] AnalysisPKG [Produce_component]
itsSimulation_station:Simula
itsCNC_milling:CNC milling
itsQuality_control_:Quality control
raw_materials
Preparation for production
raw_materials
raw_materials
Machining materials
raw_materials
components
components
Control quality
approved_components
Components
«satisfy»

«Requirement»
1722: geometry precision
ID = 1722
The manufacturing shall produce geometry with a high tolerance precision

«Requirement»
1738: precise machineries
ID = 1738
The manufacturing shall use precise machineries

«Requirement»
1728: control quality
ID = 1728
The manufacturing shall be able to control the quality of the products produced

Figure 46: Produce component activity diagram showing the link between requirements and actions.

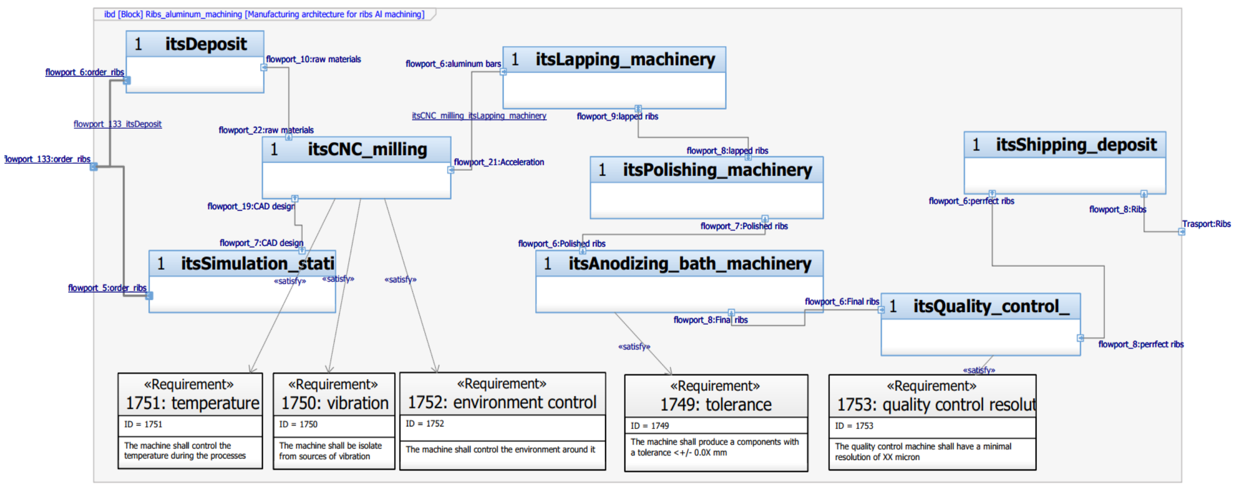

ibd [Block] Ribs_aluminum_machining [Manufacturing architecture for ribs Al machining]
1 itsDeposit
flowport_10:raw materials
flowport_6:order_ribs
flowport_133_itsDeposit
flowport_133:order_ribs
flowport_22:raw materials
1 itsCNC_milling
flowport_19:CAD design
flowport_7:CAD design
1 itsSimulation_stati
flowport_5:order_ribs
flowport_6:aluminum bars
1 itsLapping_machinery
itsCNC_milling_itsLapping_machinery
flowport_9:lapped ribs
flowport_21:Acceleration
flowport_8:lapped ribs
1 itsPolishing_machinery
flowport_7:Polished ribs
flowport_6:Polished ribs
1 itsAnodizing_bath_machinery
flowport_8:Final ribs
flowport_6:Final ribs
1 itsQuality_control_
flowport_8:perrfect ribs
1 itsShipping_deposit
flowport_6:perrfect ribs
flowport_8:Ribs
Trasport:Ribs
«satisfy»
«satisfy»
«satisfy»
«satisfy»
«satisfy»
«Requirement» 1751: temperature
ID = 1751
The machine shall control the temperature during the processes
«Requirement» 1750: vibration
ID = 1750
The machine shall be isolate from sources of vibration
«Requirement» 1752: environment control
ID = 1752
The machine shall control the environment around it
«Requirement» 1749: tolerance
ID = 1749
The machine shall produce a components with a tolerance <+/- 0.0X mm
«Requirement» 1753: quality control resolut
ID = 1753
The quality control machine shall have a minimal resolution of XX micron

Figure 47: IBD of manufacturing aluminum machining ribs showing the link between actions and components and performance requirements

These three examples show how the methodology is applied for the system under consideration. Once the basic architecture is complete the variability is introduced.

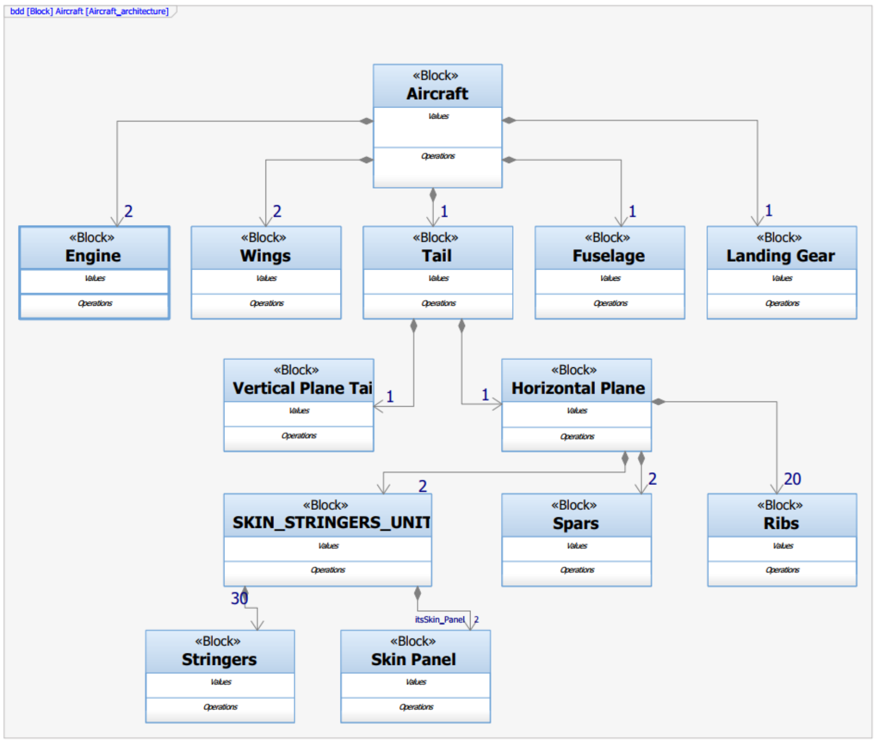


Figure 48: Aircraft System Architecture

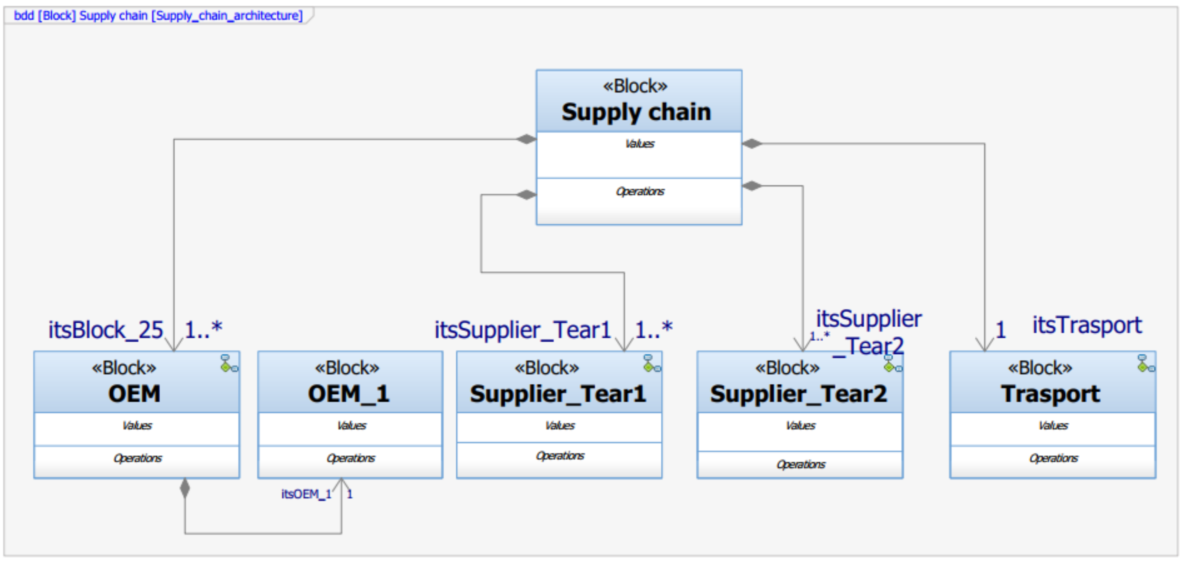


Figure 49: Supply Chain System Architecture

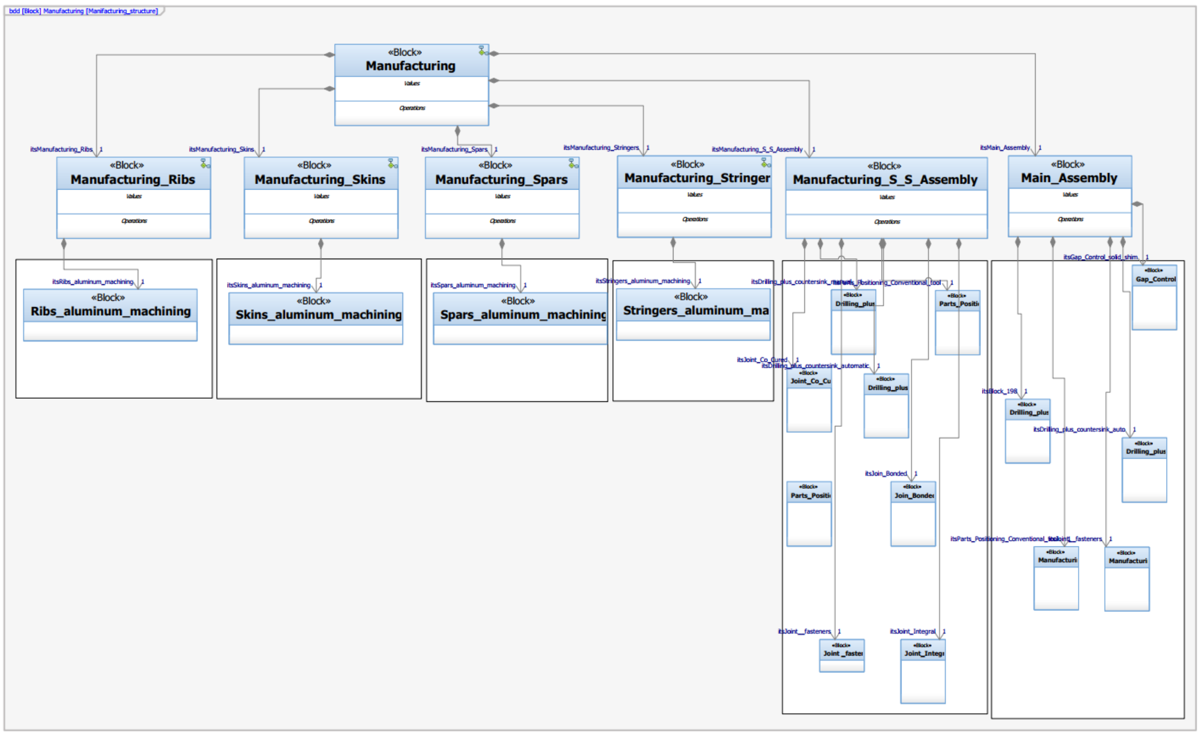

bdd [Block] Manufacturing [Manifacturing_structure]
«Block» Manufacturing
«Block» Manufacturing_Ribs
«Block» Manufacturing_Skins
«Block» Manufacturing_Spars
«Block» Manufacturing_Stringer
«Block» Manufacturing_S_S_Assembly
«Block» Main_Assembly
«Block» Ribs_aluminum_machining
«Block» Skins_aluminum_machining
«Block» Spars_aluminum_machining
«Block» Stringers_aluminum_ma
«Block» Drilling_plus
«Block» Parts_Positi
«Block» Gap_Control
«Block» Joint_Co_Cu
«Block» Join_Bonde
«Block» Manufacturi
«Block» Joint_faste
«Block» Joint_Integ

Figure 50: Manufacturing System Architecture

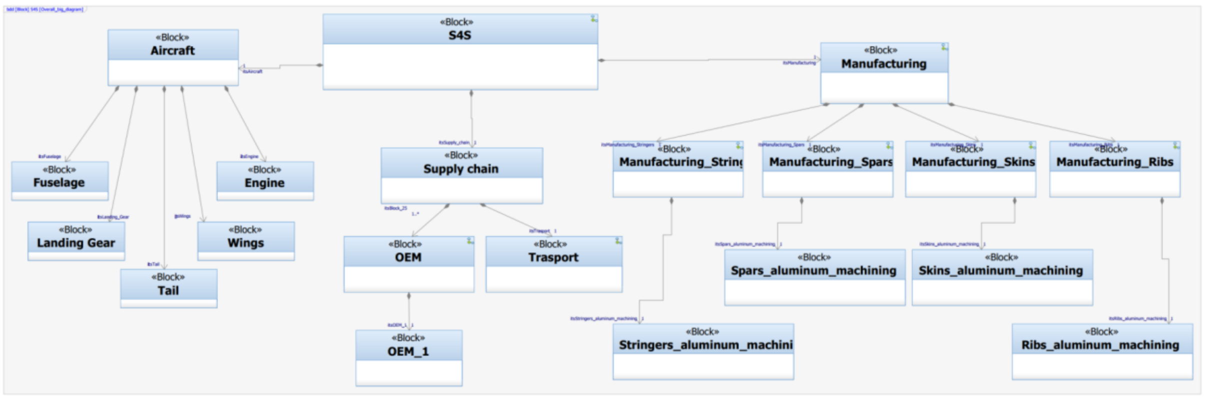


Figure 51: Basic architecture in which the behavior of the system is modeled.

5.4 150% model and variations

As introduced in the previous section, the basic architecture of the system has been found. In this section, the other variable elements of the supply chain and manufacturing system are added.

In the basic architecture, OEM site 1 is responsible for the manufacturing of all components. The material used for all components is aluminum and the process is aluminum machining. In the test case under consideration, the requirements do not vary and neither do the system functions, but only the blocks that perform those functions. The first source of variability to be considered concerns the supply chain. To achieve the 150% model, all facilities, five sites for OEM, seven for Supplier Tier 1, and nine for Supplier Tier 2, are part of the supply chain. Therefore, the supply chain's BDD is modified by considering the new blocks representing the different enterprises. The IBD of the system also changes and the connections between the various ports within the supply chain system explode in number. Figure 52 below shows how the two diagrams vary from the basic architecture.

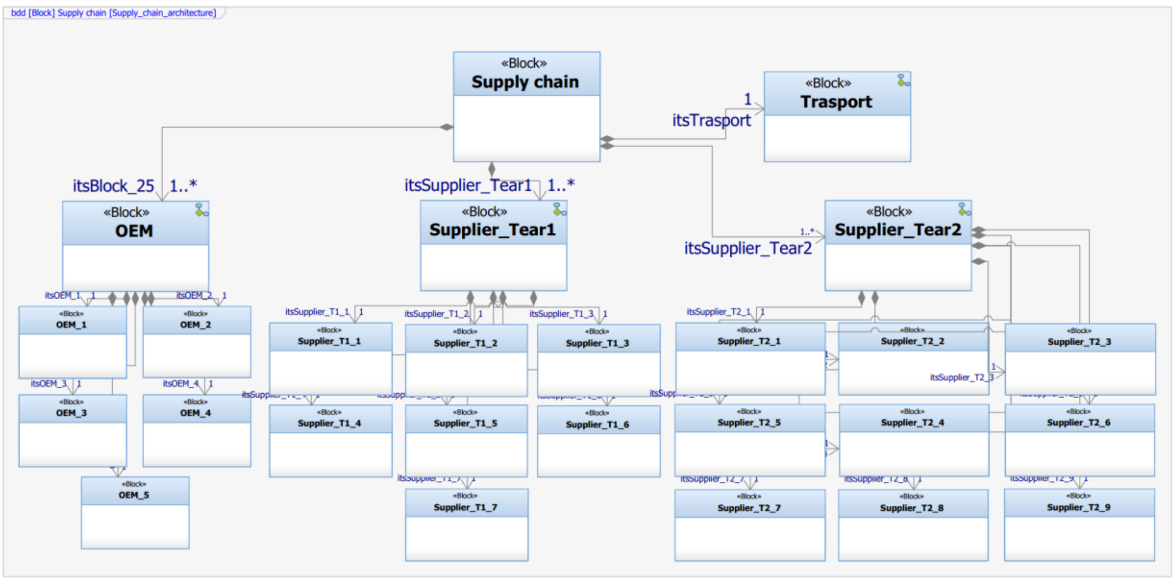
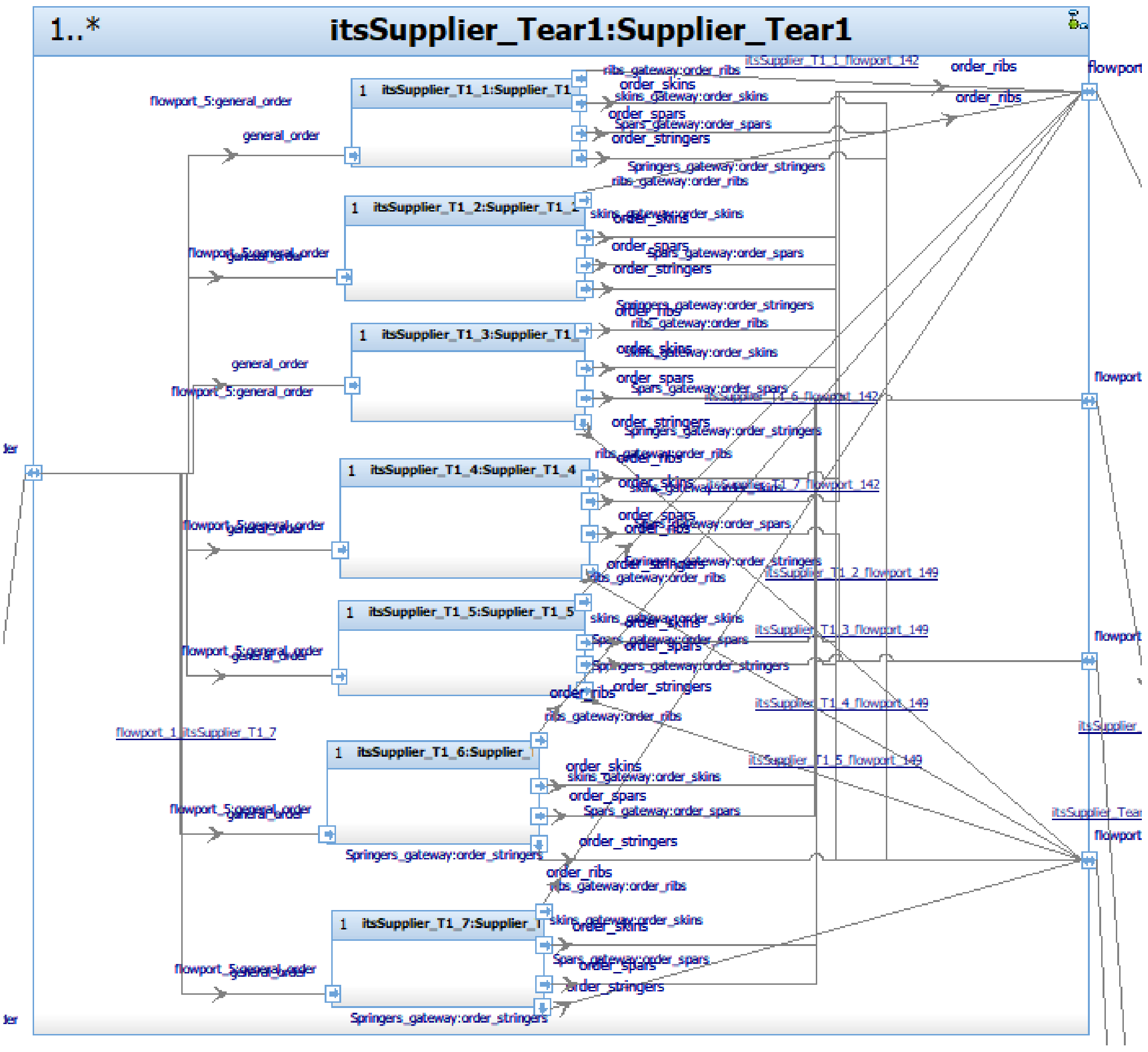


Figure 52: Supply chain structure considering all possibilities


1..*
itsSupplier_Tear1:Supplier_Tear1
1 itsSupplier_T1_1:Supplier_T1
1 itsSupplier_T1_2:Supplier_T1_2
1 itsSupplier_T1_3:Supplier_T1
1 itsSupplier_T1_4:Supplier_T1_4
1 itsSupplier_T1_5:Supplier_T1_5
1 itsSupplier_T1_6:Supplier_
1 itsSupplier_T1_7:Supplier_
flowport_5:general_order
general_order
order_ribs
order_skins
order_spars
order_stringers
ribs_gateway:order_ribs
skins_gateway:order_skins
Spars_gateway:order_spars
Springers_gateway:order_stringers
flowport
itsSupplier_T1_1_flowport_142
itsSupplier_T1_6_flowport_142
itsSupplier_T1_7_flowport_142
itsSupplier_T1_2_flowport_149
itsSupplier_T1_3_flowport_149
itsSupplier_T1_4_flowport_149
itsSupplier_T1_5_flowport_149
flowport_1_itsSupplier_T1_7

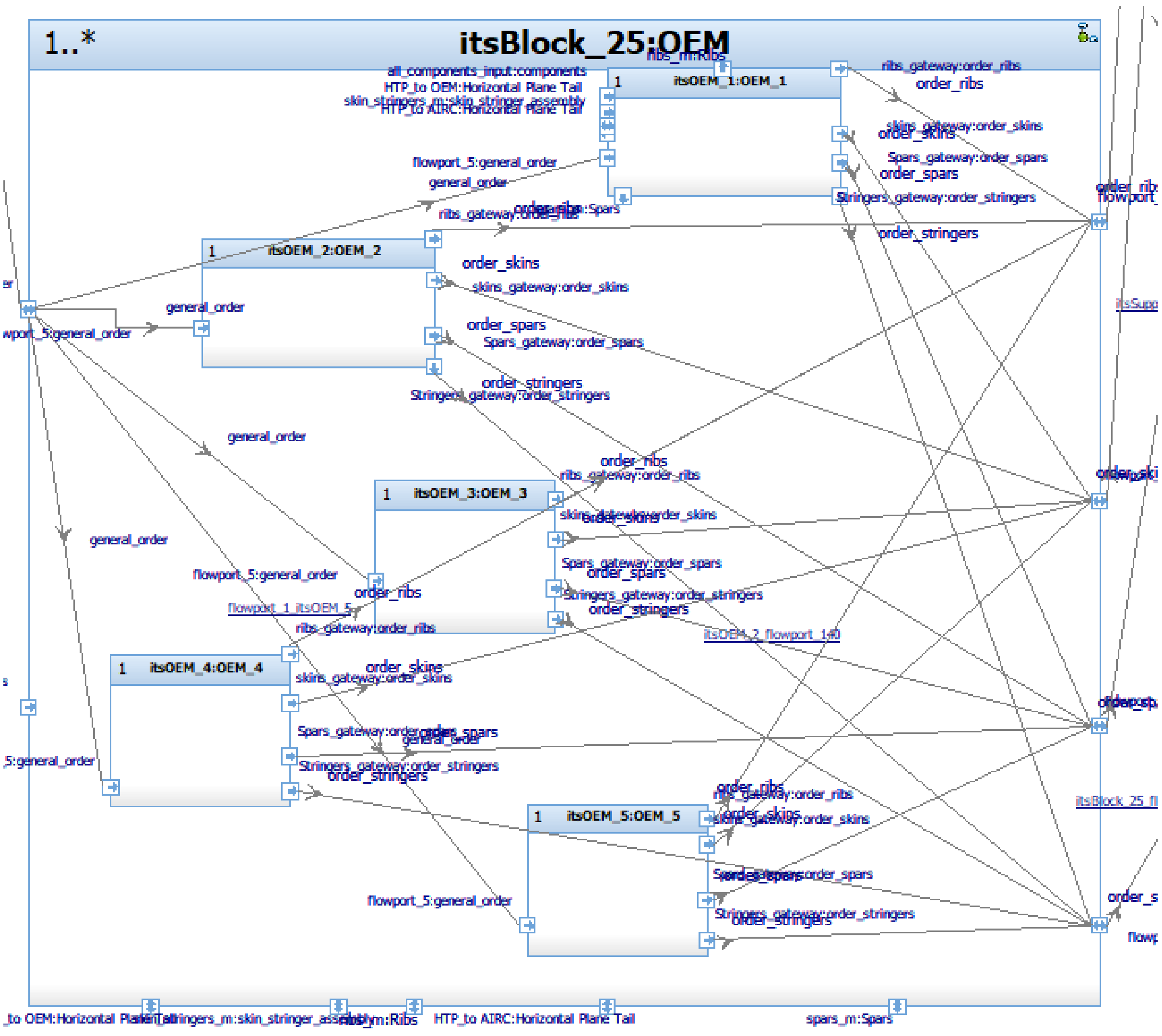


Figure 53: Example of IBD for the supply chain

The same is to be done with the processes in the manufacturing system. In which the processes selected for each component vary. The material change is not visible within the IBD s4s as the materials are characterized within each process. Selecting the process to be used automatically selects the material to be utilized. Figure 54 shows the BDD of the manufacturing system.

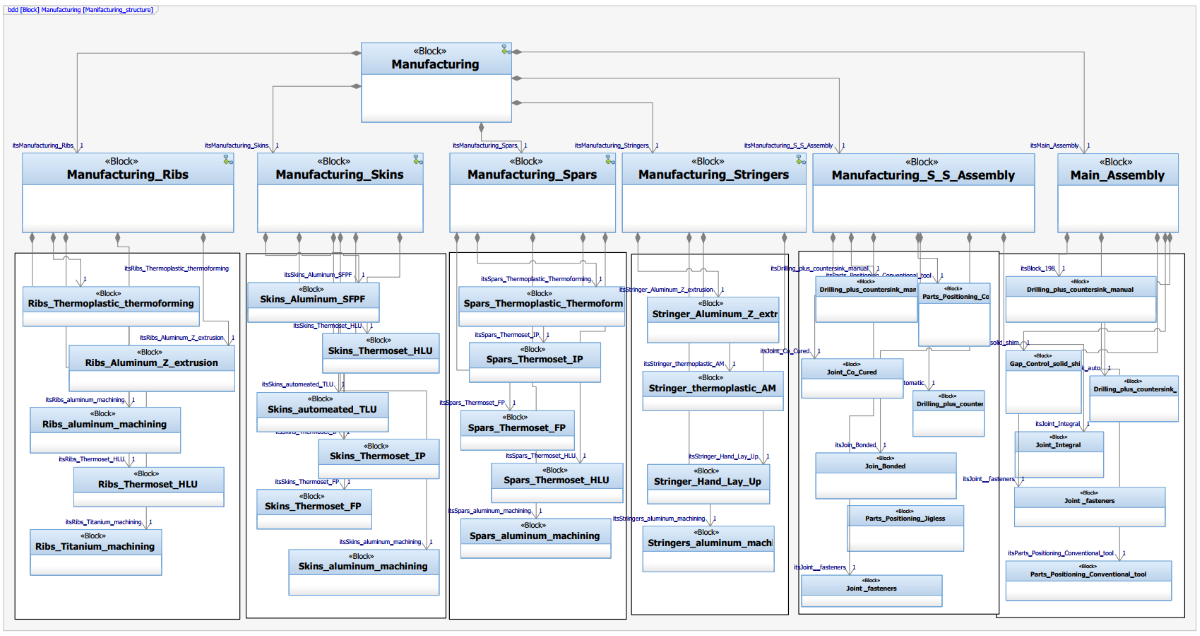
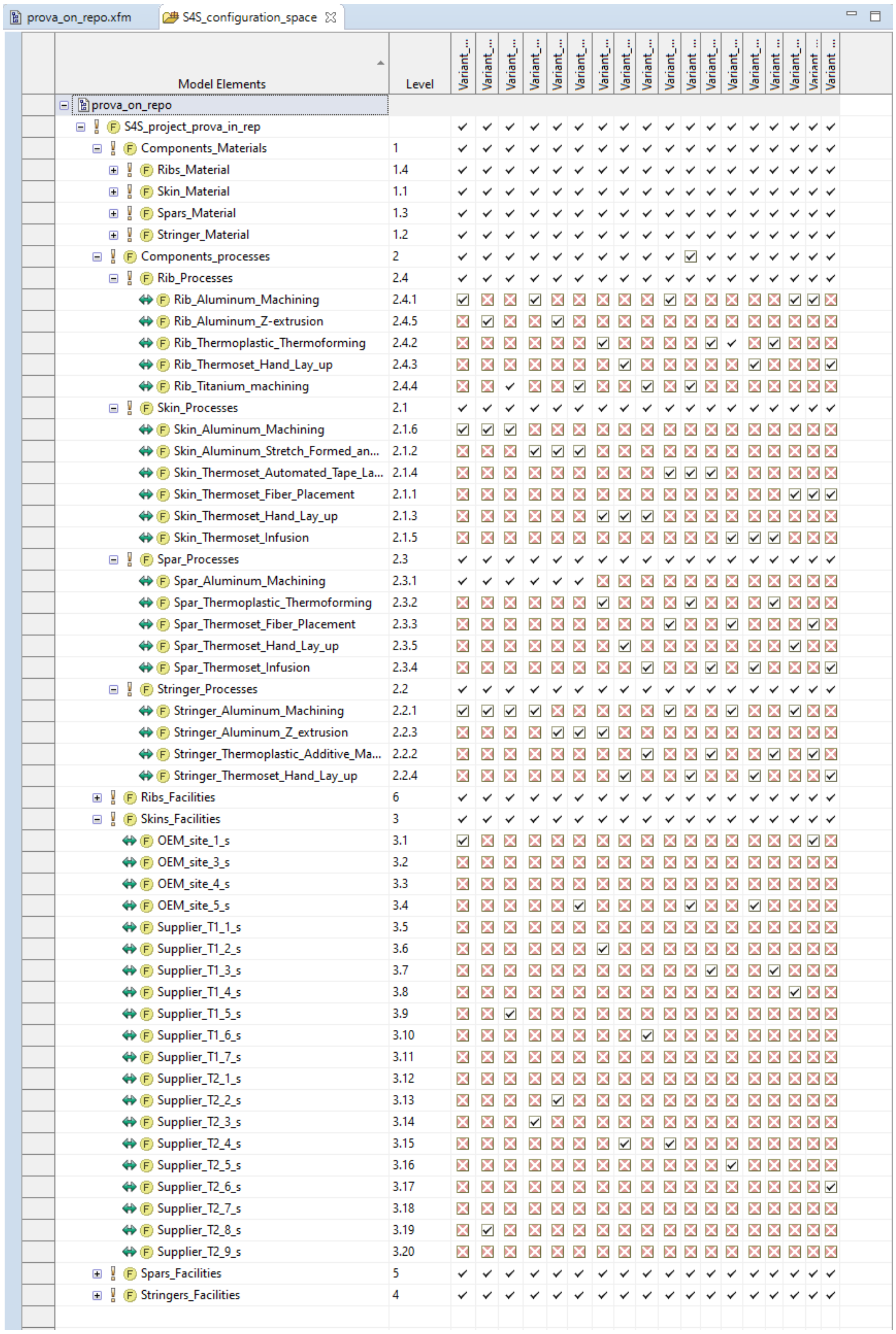


Figure 54: Manufacturing system BDD considering all processes

The 150% model is thus achieved. To derive N system variables, the feature model must be modeled. The model is constructed, using the two relationships introduced in section 3.5, by selecting the process for each component automatically the material is selected and the enterprises that cannot carry out the process are eliminated. Figure 55 shows the model.

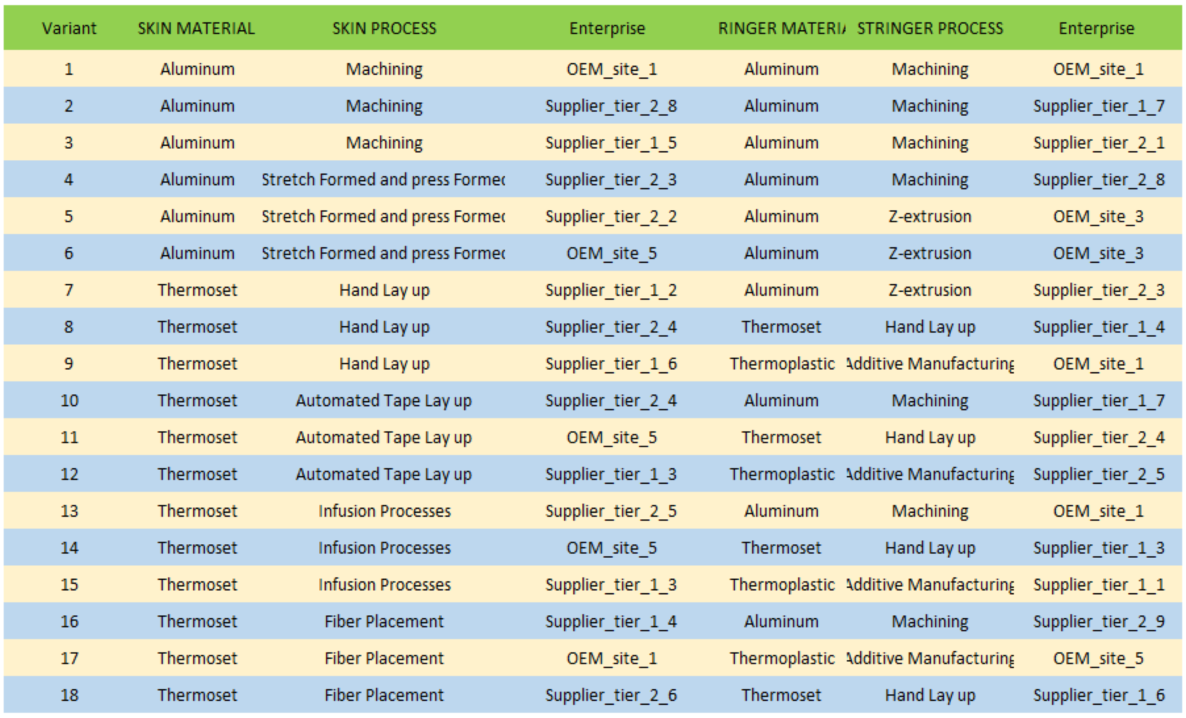

prova_on_repo.xfm | S4S_configuration_space

Model Elements	Level	Variant_...	Variant_...	Variant_...	Variant_...	Variant_...	Variant_...	Variant_...	Variant_...	Variant_...	Variant_...	Variant ...	Variant_...	Variant_...	Variant_...	Variant ...	Variant_...	Variant	Variant ...
prova_on_repo																			
S4S_project_prova_in_rep		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Components_Materials	1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Ribs_Material	1.4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Skin_Material	1.1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Spars_Material	1.3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Stringer_Material	1.2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Components_processes	2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	☑	✓	✓	✓	✓	✓	✓	✓
Rib_Processes	2.4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Rib_Aluminum_Machining	2.4.1	☑	☒	☒	☑	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☒	☑	☑	☒
Rib_Aluminum_Z-extrusion	2.4.5	☒	☑	☒	☒	☑	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Rib_Thermoplastic_Thermoforming	2.4.2	☒	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☑	✓	☒	☑	☒	☒	☒
Rib_Thermoset_Hand_Lay_up	2.4.3	☒	☒	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☒	☑	☒	☒	☒	☑
Rib_Titanium_machining	2.4.4	☒	☒	✓	☒	☒	☑	☒	☒	☑	☒	☑	☒	☒	☒	☒	☒	☒	☒
Skin_Processes	2.1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Skin_Aluminum_Machining	2.1.6	☑	☑	☑	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Skin_Aluminum_Stretch_Formed_an...	2.1.2	☒	☒	☒	☑	☑	☑	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Skin_Thermoset_Automated_Tape_La...	2.1.4	☒	☒	☒	☒	☒	☒	☒	☒	☒	☑	☑	☑	☒	☒	☒	☒	☒	☒
Skin_Thermoset_Fiber_Placement	2.1.1	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☑	☑	☑
Skin_Thermoset_Hand_Lay_up	2.1.3	☒	☒	☒	☒	☒	☒	☑	☑	☑	☒	☒	☒	☒	☒	☒	☒	☒	☒
Skin_Thermoset_Infusion	2.1.5	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☑	☑	☑	☒	☒	☒
Spar_Processes	2.3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Spar_Aluminum_Machining	2.3.1	✓	✓	✓	✓	✓	✓	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Spar_Thermoplastic_Thermoforming	2.3.2	☒	☒	☒	☒	☒	☒	☑	☒	☒	☒	☑	☒	☒	☒	☑	☒	☒	☒
Spar_Thermoset_Fiber_Placement	2.3.3	☒	☒	☒	☒	☒	☒	☒	☒	☒	☑	☒	☒	☑	☒	☒	☒	☑	☒
Spar_Thermoset_Hand_Lay_up	2.3.5	☒	☒	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☒	☒	☒	☑	☒	☒
Spar_Thermoset_Infusion	2.3.4	☒	☒	☒	☒	☒	☒	☒	☒	☑	☒	☒	☑	☒	☑	☒	☒	☒	☑
Stringer_Processes	2.2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Stringer_Aluminum_Machining	2.2.1	☑	☑	☑	☑	☒	☒	☒	☒	☒	☑	☒	☒	☑	☒	☒	☑	☒	☒
Stringer_Aluminum_Z_extrusion	2.2.3	☒	☒	☒	☒	☑	☑	☑	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Stringer_Thermoplastic_Additive_Ma...	2.2.2	☒	☒	☒	☒	☒	☒	☒	☒	☑	☒	☒	☑	☒	☒	☑	☒	☑	☒
Stringer_Thermoset_Hand_Lay_up	2.2.4	☒	☒	☒	☒	☒	☒	☒	☑	☒	☒	☑	☒	☒	☑	☒	☒	☒	☑
Ribs_Facilities	6	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Skins_Facilities	3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
OEM_site_1_s	3.1	☑	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☑	☒
OEM_site_3_s	3.2	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
OEM_site_4_s	3.3	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
OEM_site_5_s	3.4	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☑	☒	☒	☑	☒	☒	☒	☒
Supplier_T1_1_s	3.5	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Supplier_T1_2_s	3.6	☒	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Supplier_T1_3_s	3.7	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☑	☒	☒	☑	☒	☒	☒
Supplier_T1_4_s	3.8	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☑	☒	☒
Supplier_T1_5_s	3.9	☒	☒	☑	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Supplier_T1_6_s	3.10	☒	☒	☒	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☒	☒	☒	☒	☒
Supplier_T1_7_s	3.11	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Supplier_T2_1_s	3.12	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Supplier_T2_2_s	3.13	☒	☒	☒	☒	☑	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Supplier_T2_3_s	3.14	☒	☒	☒	☑	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Supplier_T2_4_s	3.15	☒	☒	☒	☒	☒	☒	☒	☑	☒	☑	☒	☒	☒	☒	☒	☒	☒	☒
Supplier_T2_5_s	3.16	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☒
Supplier_T2_6_s	3.17	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☑
Supplier_T2_7_s	3.18	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Supplier_T2_8_s	3.19	☒	☑	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Supplier_T2_9_s	3.20	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Spars_Facilities	5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Stringers_Facilities	4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

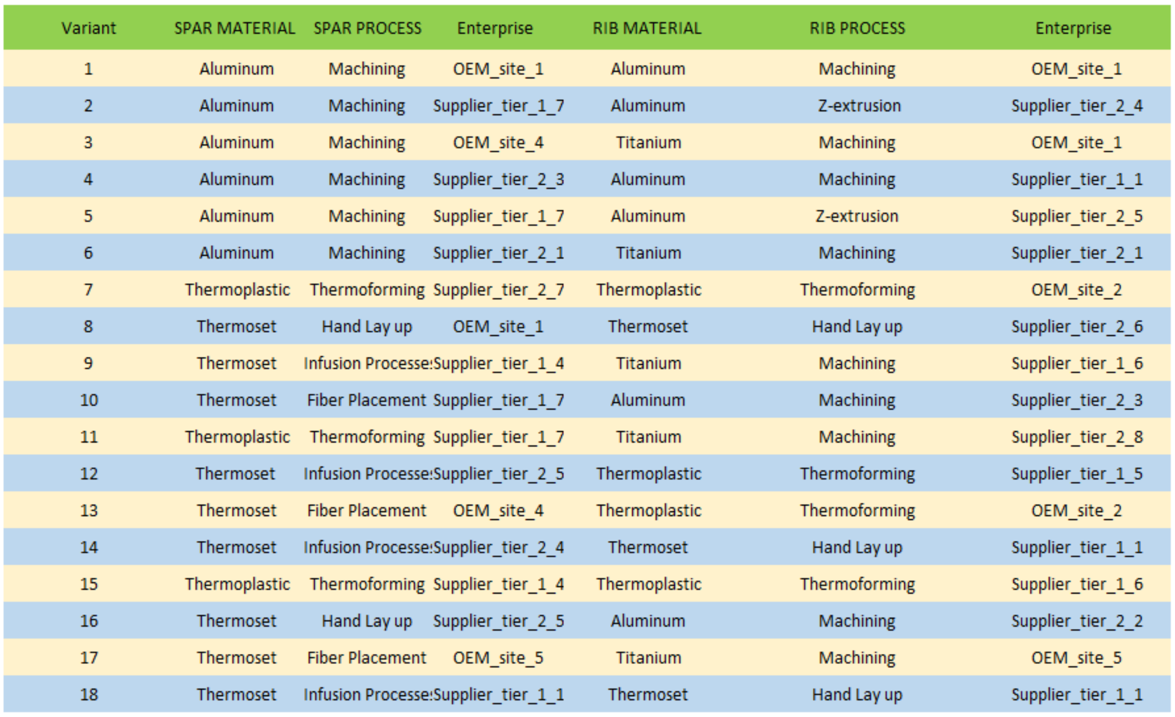
Figure 55: Feature Model and the 18 variants selected for the test case

However, the feature model can also be used in a more classical manner in which first the material is selected, then the process and finally the enterprise. The use of relations still allows a possible structure to be arrived at from any given feature. If an incorrect feature is selected, the model automatically deselects the error.

For the case under consideration, thousands of solutions are possible but only 18 are considered. The only care taken in the selection is that all processes and materials are selected at least once and that at least one variant of the system shows a supply chain consisting only of OEMs and one variant in which the supply chain consists only of suppliers. Figure 56 shows the different variants.



Variant	SKIN MATERIAL	SKIN PROCESS	Enterprise	RINGER MATERI	STRINGER PROCESS	Enterprise
1	Aluminum	Machining	OEM_site_1	Aluminum	Machining	OEM_site_1
2	Aluminum	Machining	Supplier_tier_2_8	Aluminum	Machining	Supplier_tier_1_7
3	Aluminum	Machining	Supplier_tier_1_5	Aluminum	Machining	Supplier_tier_2_1
4	Aluminum	Stretch Formed and press Formed	Supplier_tier_2_3	Aluminum	Machining	Supplier_tier_2_8
5	Aluminum	Stretch Formed and press Formed	Supplier_tier_2_2	Aluminum	Z-extrusion	OEM_site_3
6	Aluminum	Stretch Formed and press Formed	OEM_site_5	Aluminum	Z-extrusion	OEM_site_3
7	Thermoset	Hand Lay up	Supplier_tier_1_2	Aluminum	Z-extrusion	Supplier_tier_2_3
8	Thermoset	Hand Lay up	Supplier_tier_2_4	Thermoset	Hand Lay up	Supplier_tier_1_4
9	Thermoset	Hand Lay up	Supplier_tier_1_6	Thermoplastic	Additive Manufacturing	OEM_site_1
10	Thermoset	Automated Tape Lay up	Supplier_tier_2_4	Aluminum	Machining	Supplier_tier_1_7
11	Thermoset	Automated Tape Lay up	OEM_site_5	Thermoset	Hand Lay up	Supplier_tier_2_4
12	Thermoset	Automated Tape Lay up	Supplier_tier_1_3	Thermoplastic	Additive Manufacturing	Supplier_tier_2_5
13	Thermoset	Infusion Processes	Supplier_tier_2_5	Aluminum	Machining	OEM_site_1
14	Thermoset	Infusion Processes	OEM_site_5	Thermoset	Hand Lay up	Supplier_tier_1_3
15	Thermoset	Infusion Processes	Supplier_tier_1_3	Thermoplastic	Additive Manufacturing	Supplier_tier_1_1
16	Thermoset	Fiber Placement	Supplier_tier_1_4	Aluminum	Machining	Supplier_tier_2_9
17	Thermoset	Fiber Placement	OEM_site_1	Thermoplastic	Additive Manufacturing	OEM_site_5
18	Thermoset	Fiber Placement	Supplier_tier_2_6	Thermoset	Hand Lay up	Supplier_tier_1_6



Variant	SPAR MATERIAL	SPAR PROCESS	Enterprise	RIB MATERIAL	RIB PROCESS	Enterprise
1	Aluminum	Machining	OEM_site_1	Aluminum	Machining	OEM_site_1
2	Aluminum	Machining	Supplier_tier_1_7	Aluminum	Z-extrusion	Supplier_tier_2_4
3	Aluminum	Machining	OEM_site_4	Titanium	Machining	OEM_site_1
4	Aluminum	Machining	Supplier_tier_2_3	Aluminum	Machining	Supplier_tier_1_1
5	Aluminum	Machining	Supplier_tier_1_7	Aluminum	Z-extrusion	Supplier_tier_2_5
6	Aluminum	Machining	Supplier_tier_2_1	Titanium	Machining	Supplier_tier_2_1
7	Thermoplastic	Thermoforming	Supplier_tier_2_7	Thermoplastic	Thermoforming	OEM_site_2
8	Thermoset	Hand Lay up	OEM_site_1	Thermoset	Hand Lay up	Supplier_tier_2_6
9	Thermoset	Infusion Processe	Supplier_tier_1_4	Titanium	Machining	Supplier_tier_1_6
10	Thermoset	Fiber Placement	Supplier_tier_1_7	Aluminum	Machining	Supplier_tier_2_3
11	Thermoplastic	Thermoforming	Supplier_tier_1_7	Titanium	Machining	Supplier_tier_2_8
12	Thermoset	Infusion Processe	Supplier_tier_2_5	Thermoplastic	Thermoforming	Supplier_tier_1_5
13	Thermoset	Fiber Placement	OEM_site_4	Thermoplastic	Thermoforming	OEM_site_2
14	Thermoset	Infusion Processe	Supplier_tier_2_4	Thermoset	Hand Lay up	Supplier_tier_1_1
15	Thermoplastic	Thermoforming	Supplier_tier_1_4	Thermoplastic	Thermoforming	Supplier_tier_1_6
16	Thermoset	Hand Lay up	Supplier_tier_2_5	Aluminum	Machining	Supplier_tier_2_2
17	Thermoset	Fiber Placement	OEM_site_5	Titanium	Machining	OEM_site_5
18	Thermoset	Infusion Processe	Supplier_tier_1_1	Thermoset	Hand Lay up	Supplier_tier_1_1

Figure 56: The 18 variants selected to be evaluated

Before the variants of the architecture can be displayed, each feature must be connected to the variable elements of the model 150%. The connection must be made with the blocks in the BDD representing the supply chain facilities and the connectors inside the IBD representing the information flows. This is achieved using the Pure::Variants integration tool in IBM Rhapsody, which precisely allows the variants to be connected to the system elements and the desired variant model to be selected.

Following the selection of the variant model, the elements connected to the non-selected features are removed from the architecture. One example of an architecture variant is presented in Figure 57: Model 150% after the Variant Model is applied.

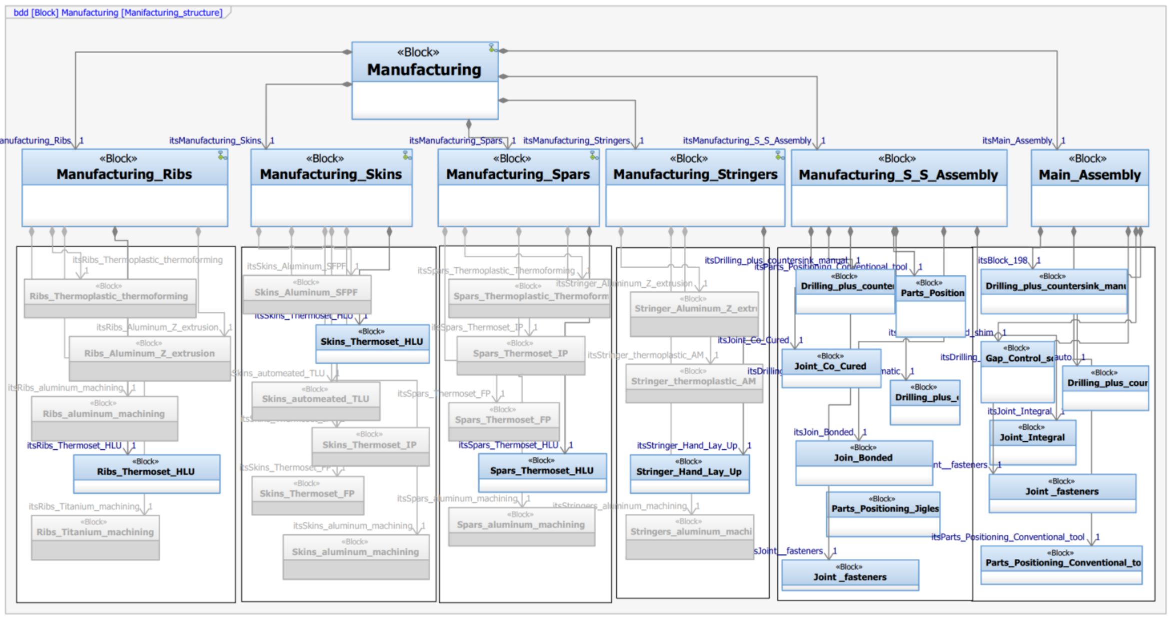

bdd [Block] Manufacturing [Manifacturing_structure]
«Block» Manufacturing
«Block» Manufacturing_Ribs
«Block» Manufacturing_Skins
«Block» Manufacturing_Spars
«Block» Manufacturing_Stringers
«Block» Manufacturing_S_S_Assembly
«Block» Main_Assembly
Ribs_Thermoplastic_thermoforming
Ribs_Aluminum_Z_extrusion
Ribs_aluminum_machining
Ribs_Thermoset_HLU
Ribs_Titanium_machining
Skins_Aluminum_SFPF
Skins_Thermoset_HLU
Skins_automeated_TLU
Skins_Thermoset_IP
Skins_Thermoset_FP
Skins_aluminum_machining
Spars_Thermoplastic_Thermoforming
Spars_Thermoset_IP
Spars_Thermoset_FP
Spars_Thermoset_HLU
Spars_aluminum_machining
Stringer_Aluminum_Z_extrusion
Stringer_thermoplastic_AM
Stringer_Hand_Lay_Up
Stringers_aluminum_machining
Drilling_plus_countersink
Parts_Position
Joint_Co_Cured
Drilling_plus
Join_Bonded
Parts_Positioning_Jigles
Joint_fasteners
Drilling_plus_countersink_manual
Gap_Control_shim
Drilling_plus_countersink_auto
Joint_Integral
Joint_fasteners
Parts_Positioning_Conventional_tool

Figure 57: Model 150% after the Variant Model is applied

In order to derive all architecture, it is no longer necessary to connect features to elements, but to repeat the variant model selection procedure.

Once all architectures are derived, they can be analyzed using the methodology presented in Section 3.6.

5.5 MAUT analysis

Depending on the variable to be selected, the workflow input file varies. This is why 18 different files are created, the only thing that varies within the CPACS file input are the vectors that refer to the processes, and thus the materials selected for the variant. Each process is associated with a vector containing twenty-one elements, one per facility. The vector is null if that process is unused in the selected variant, it has only one non-null component, located at the position that identifies the facility used, to indicate that the process is used. The first five positions of the vector are for OEM, the next seven for Supplier Tier 1, and the last nine for Supplier Tier 2. At the end of the workflow, the data is retrieved from the output CPACS file and uploaded to Excel to analyze the results. Five diagrams are shown below, the first is the value associated with the fuel consumption of the aircraft as a function of cost. The next three diagrams are the values concerning the supply chain as a function of cost (risk, quality, and time), and finally the last one shows the total value as a function of cost. The variants with the best fuel consumption risk, quality, and time are highlighted respectively by black, green, red, and blue for better visualization.

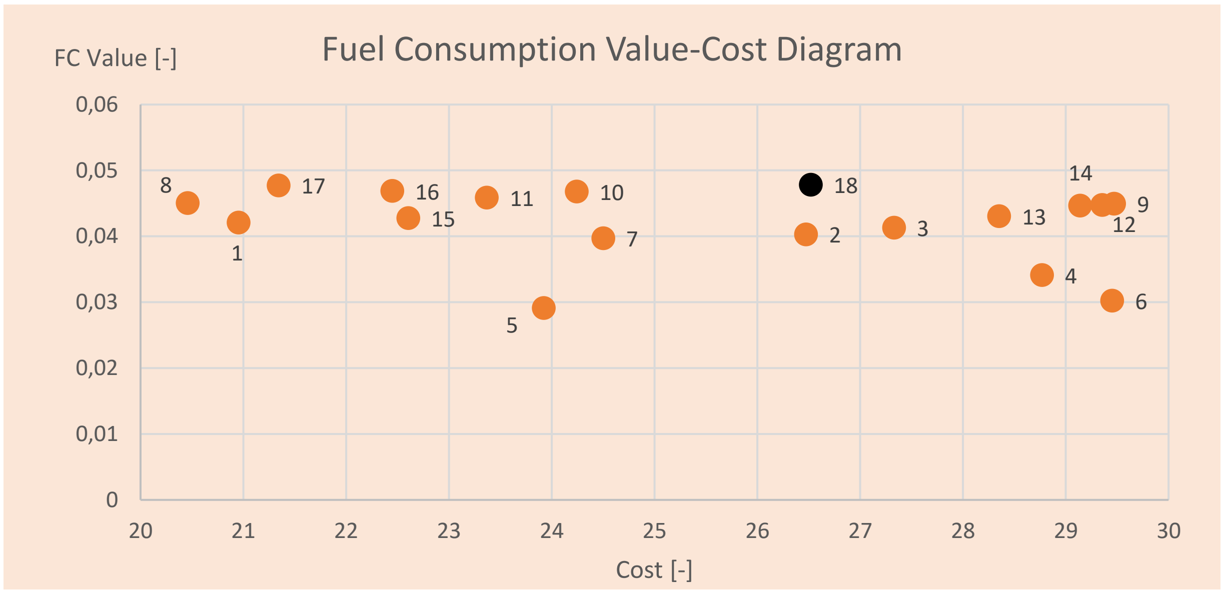


Figure 58: FC Value-Cost Diagram

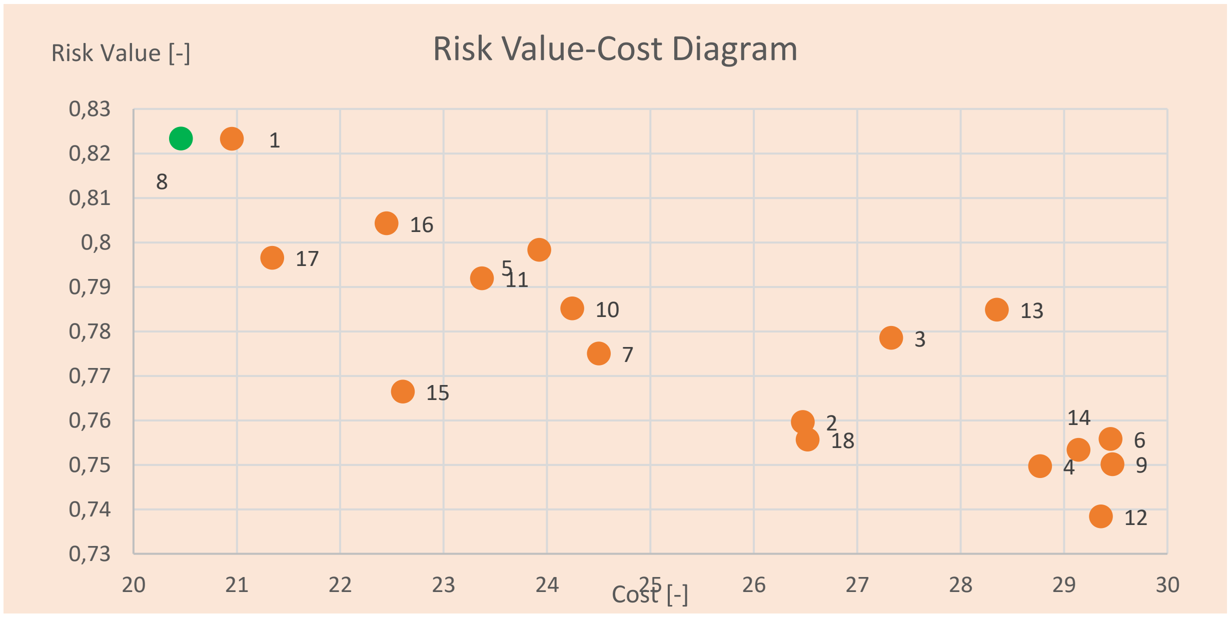


Figure 59: Risk Value-Cost Diagram

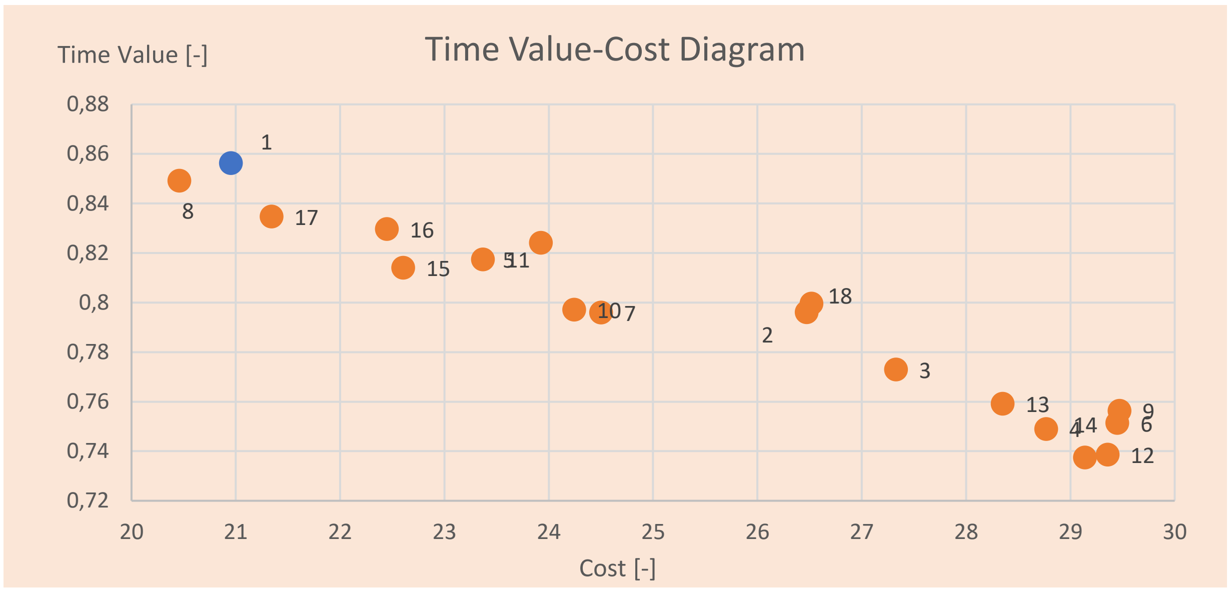


Figure 60: Time Value-Cost Diagram

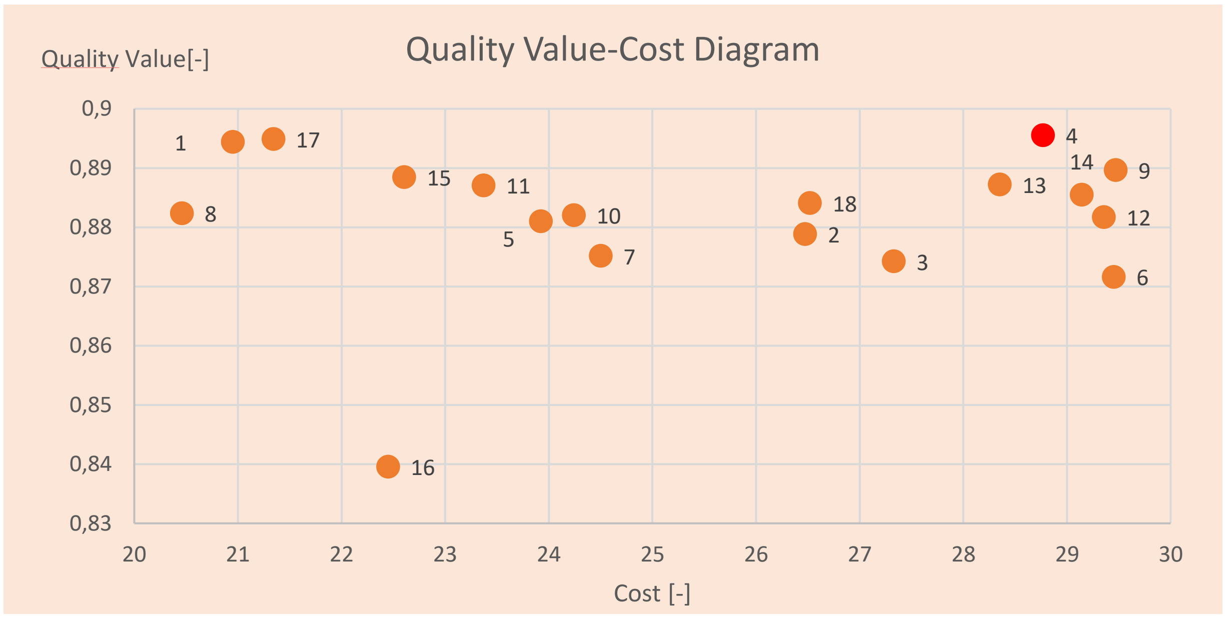


Figure 61: Quality Value-Cost Diagram

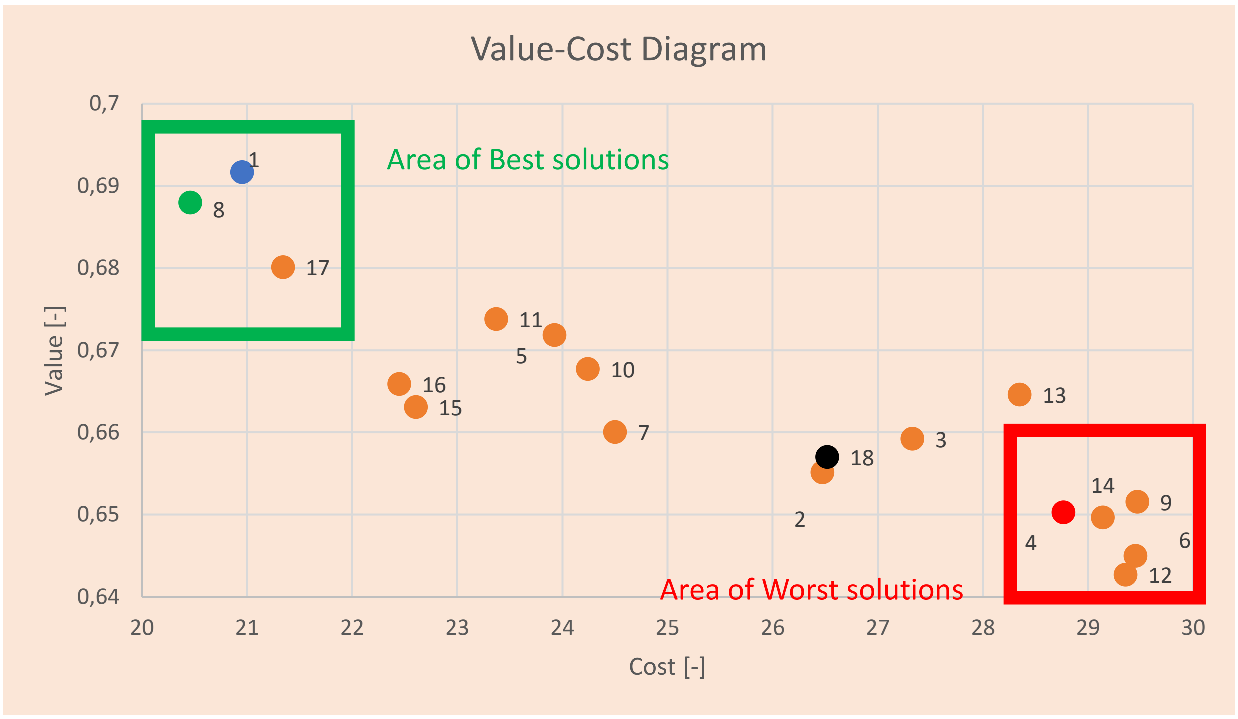


Figure 62: Value-Cost Diagram

In the eighteen cases analyzed, solution 1, in which all components are made of aluminum by machining and all processes are carried out by OEM site 1, is the one with the highest value. From the data available the OEM has high skills, and this translates into low risk, high quality, and low time plus having assumed all assembly done by this facility the transport time and costs are zero helping. For these reasons, these results were expected, with variant number 1 being the best solution for the model used. However, it should be noted that other solutions can be considered. Solution 17, in which all processes are carried out by the OEM but in different plants presents a higher quality and fuel consumption value, stakeholders might opt for this solution if they value these factors more than others. Solution 8 is also interesting and has a similar total value to the first solution with higher risk and fuel consumption.

6 Conclusion

This section summarizes the results found and suggests activities to be addressed to deepen the research.

6.1 Summary

The proposed thesis discloses a methodology, pertinent to the field of systems engineering, to be applied to the conceptual design phase of a system. The aim is to reduce the costs, design, and production times of an aeronautical system and its environmental impact.

To enable this, the methodology presents two innovations:

- A concurrent design phase extended to three systems: System of Interest and two systems related to the production of the system components, namely supply chain and manufacturing. This approach contrasts with the traditional sequential approach.
- Secondly, through the joint use of MBSE and PLE-related dictates, combining them to quickly and efficiently generate the system architecture according to stakeholder requirements and also generating the variables to be analyzed to provide as much information as possible to make a considered choice.

The conceptual design phase of the three systems considered, aircraft, manufacturing, and supply chain, follows the five processes depicted in Figure 9. Of these five, the methodology touches on four, namely <u>System Specifications</u>, <u>System Architecture</u>, <u>System synthesis, and exploration.</u>

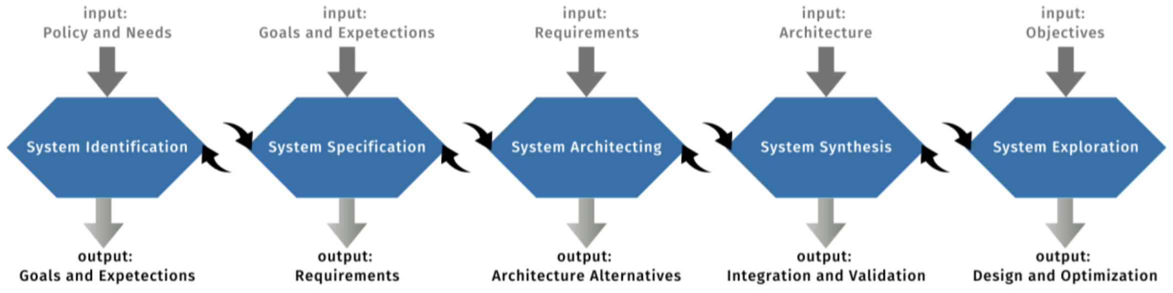


Figure 9: Processes included in the S4S framework

- For requirements, the standard is followed, the spectrum of stakeholders to be considered is not limited to SoI, but also embraces the two enabling systems. Thus, not only are the goals, wishes, limits, and constraints of the SoI considered but also those of the other systems. This results in unfeasible solutions being automatically excluded and also tightens the relationship between the stakeholders of the three systems.
- Following the stipulation of the requirements, the architecture comprising the three systems is identified. To search for it, the requirements were considered as a pivot for development: first by considering functional requirements to search for high-level functions that the system must perform, then through lower-level requirements to identify more specific behaviors to be performed by components or subsystems. Finally, through performance requirements or design constraints select components with the required performance. Once the basic architecture is reached, it is updated by considering all the variable elements of the system, reaching what is called the 150 percent model.
- After that, variants are generated using the PLE approach. A feature model with all variable features is modeled, and variant models of the different architectural variants of the system are generated and bound to the model 150%. The result is N 100% models.
- Finally, the system variants are evaluated considering product and production performance. This is done by considering several criteria, weighed and aggregated into a single value, which is then plotted in a dimensionless value-cost space.

To test the soundness of the methodology, a case study is considered, in which the product developed, the horizontal tail plane of an aircraft, its supply chain, and the manufacturing of its components are taken into account. The four components are ribs, spars, stringers, and skin panels and constitute all the structural elements of the system considered. Variability is added by considering a multitude of different materials, processes, and facilities. Of the thousands of possible solutions, eighteen are analyzed through a MAUT analysis. To evaluate the best solution, the competencies and position of the companies are taken into account, as well as the fuel consumption of the aircraft.

6.2 Finding and next steps

The selected case study made it possible to verify the present methodology. In fact, the results showed not only variants that meet all stakeholder requirements, but also variants that, respecting the constraints imposed by supply chain and manufacturing, should not be delayed due to redesign or process infeasibility. Consequently, once the most suitable variant has been chosen according to stakeholder objectives, it can be produced without further delay. The contribution of the PLE approach has been vital in speeding up modeling time and providing numerous variants to choose from.

However, some improvements can be considered in future research. In setting up the model, no experts from the manufacturing system were involved and, in general, little data is available on processes and production lines. Future work can address this problem by initiating conversations and collaborations with experts in the field to improve the model.

Furthermore, although the PLE approach was instrumental in speeding up the timeline, feature selection for each variant model was done manually. This constitutes a bottleneck in the process. Future work can try to solve this problem by proposing a tool or extension to automate the creation of variant models, so that all possible variants can be generated based on the relationships between features. Another possibility is to develop a methodology that allows certain solutions to be excluded upstream based on the specific knowledge of each facility.

7 References

AWS. (n.d.). *What is MBSE and why do industries start to use?* . Model Based Systems Engineering (MBSE) on AWS: From Migration to Innovation. (n.d.)..). Https://Docs.Aws.Amazon.Com/Whitepapers/Latest/Model-Based-Systems-Engineering/What-Is-Mbse-and-Why-Do-Industries-Start-to-Use.Html.

Bajaj, M., Friedenthal, S., & Seidewitz, E. (2022). Systems Modeling Language (SysML v2) Support for Digital Engineering. *INSIGHT*, *25*(1), 19–24. https://doi.org/10.1002/inst.12367

Benina, F., Benlahrache, N., Belala, F., & Kacem, A. H. (2023). *Modélisation et Analyse des Systèmes Cyber-physiques : Cas du Système de Surveillance Continue du Glucose*. http://ceur-ws.org

Boggero, L., Ciampa, P. D., & Nagel, B. (2020). *The application of the agile 4.0 MBSE architectural framework for the modeling of system stakeholders, needs and requirements*.

Box George E.P., D. N. R. (1987). *Empirical Model-Building and Response Surfaces* (Wiley, Ed.).

Broodney, H., Dotan, D., Greenberg, L., & Masin, M. (2012). 1.6.2 Generic Approach for Systems Design Optimization in MBSE1. *INCOSE International Symposium*, *22*(1), 184–200. https://doi.org/10.1002/j.2334-5837.2012.tb01330.x

Bussemaker, J. H., Boggero, L., & Nagel, B. (2024, July 29). System Architecture Design Space Exploration: Integration With Computational Environments and Efficient Optimization. *AIAA AVIATION FORUM AND ASCEND 2024*. https://doi.org/10.2514/6.2024-4647

Canis, B. (2011). *CRS Report for Congress The Motor Vehicle Supply Chain: Effects of the Japanese Earthquake and Tsunami The Motor Vehicle Supply Chain: Effects of the Japanese Earthquake and Tsunami Congressional Research Service*. www.crs.gov

Clements, P., & Northrop, L. (2001). *Software Product Lines: Practices and Patterns*. Springer.

Cloutier, R. , B. M. , & L. D. (2010). MBSE in support of complex systems engineering. *INCOSE International Symposium*, 1767–1779.

DLR. (2020). *Official RCE website. Available online at https://rcenvironment.de/.*

Donelli, G., Boggero, L., & Nagel, B. (2023b). Concurrent Value-Driven Decision-Making Process for the Aircraft, Supply Chain and Manufacturing Systems Design. *Systems*, *11*(12). https://doi.org/10.3390/systems11120578

Donelli, G., Ciampa, P. D., Mello, J. M. G., Odaguil, F. I. K., Cuco, A. P. C., & van der Laan, T. (2023b). A Value-driven Concurrent Approach for Aircraft Design-Manufacturing-Supply Chain. *Production and Manufacturing Research*, *11*(1). https://doi.org/10.1080/21693277.2023.2279709

Donelli, G., Mello, J. M. G. D., Odaguil, F. I. K., Van Der Laan, T., Boggero, L., & Nagel, B. (n.d.). *Value-driven Tradespace Exploration for Aircraft Design, Manufacturing and Supply Chain*.

Donelli Giuseppa. (2024). *Systems Engineering holistic approach for aircraft, manufacturing and supply chain concurrent design*. University of Naples.

Dutta Banik, P., & Sengupta, Dr. S. (2024, October 9). Compare and Analysis MBSE Benefits with Document-Centric Traditional System Engineering Approach. *Proceedings of the International Conference on Industrial Engineering and Operations Management*. https://doi.org/10.46254/wc01.20240082

E. Freund. (2005). *INTERNATIONAL ISO/IEC STANDARD 15288:2002 systems engineering-system life-cycle processes*.

Engineering, P. L. (2022). *What is Product Line Engineering?* Https://Productlineengineering.Com/Resources/What-Is-Product-Line-Engineering/.

Engineering Standards Committee of the IEEE Computer Society, S. (2011). *ISO/IEC/IEEE 29148:2011(E), Systems and software engineering — Life cycle processes — Requirements engineering*. www.iso.org

Estefan, J. A. (2007). Survey of model-based systems engineering (MBSE) methodologies. *Incose MBSE Focus Group, 25(8), 1-12.*

Estefan Jeffrey. (2011). Methodology and Metrics Activity: Overview, Update, and Breakout Agenda. *INCOSE International Workshop.*

Flightpath 2050. (2011). *Flightpath 2050 : Europe's vision for aviation : maintaining global leadership and serving society's needs*. Publications Office.

Forlingieri, M. (2022, February 23). The four dimensions of Variability and their impact on MBPLE: How to approach variability in the development of aircraft product lines at Airbus. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3510466.3511275

Forlingieri, M., & Weilkiens, T. (2022a). Two Variant Modeling Methods for MBPLE at Airbus. *INCOSE International Symposium*, *32*(1), 1097–1113. https://doi.org/10.1002/iis2.12984

Friedenthal, M. & S. (2014). *A Practical Guide to SysML The Systems Modeling Language.*

Gedell, S., & Johannesson, H. (2013). Design rationale and system description aspects in product platform design: Focusing reuse in the design lifecycle phase. *Concurrent Engineering*, *21*(1), 39–53. https://doi.org/10.1177/1063293X12469216

Gore, A. , L. R. , V. W. A. , & S. B. (2020). The digital twin and beyond: The evolution of MBSE in the era of smart systems. In *Journal of Systems Engineering* (Vol. 23, pp. 145–159).

Ryan, M., & Wheatcraft, L. (2022). *Guide to Writing Requirements*. http://www.incose.org

Heydari S.A. (2023). Model-Based Systems Engineering (MBSE) for Complex Engineering Projects. *Management Strategies and Engineering Sciences.*

Holt, J. , & P. S. (2010). *Modelling enterprise architectures*. Springer.

Hummell, J., & Hause, M. (2015). Model-based Product Line Engineering - enabling product families with variants. *2015 IEEE Aerospace Conference*, 1–8. https://doi.org/10.1109/AERO.2015.7119108

IBM. (2024a). *BM Engineering Requirements Management DOORS. Software for requirements management. Version 9.7: IBM Corporation. Available online at https://www.ibm.com/.*

IBM. (2024b). *IBM Engineering Systems Design Rhapsody. Software for systems and software development. Version 9.0: IBM Coroporation. Available online at https://www.ibm.com/. International.*

INCOSE. (2021). Product Line Engineering: Managing Variability and Complexity. *International Council on Systems Engineering (INCOSE).*

INCOSE. (2022). Model-Based Systems Engineering in Aerospace: The Boeing 787 Case Study. *International Council on Systems Engineering (INCOSE).*

INCOSE. (2023). *INCOSE systems engineering handbook* (John Wiley & Sons, Ed.).

ISO/IEC 26580:2021. (n.d.). *Software and systems engineering — Methods and tools for the feature-based approach to software and systems product line engineering.*

Jorgensen, R. (2011). Defining Operational Concepts using SysML: System Definition from the Human Perspective. *INCOSE International Symposium*, *21*(1), 3005–3138. https://doi.org/10.1002/j.2334-5837.2011.tb01307.x

J.S. Topper N.C. Horner. (2013). Model-Based Systems Engineering in Support of Complex Systems Development. *Johns Hopkins APL Technical Digest*, *32*, 419–432.

London, B. (2011). *A Model-Based Systems Engineering Framework for Concept Development*. MIT.

Madeira, R. H., de Sousa Pinto, D. H., & Forlingieri, M. (2023). Variability on System Architecture using Airbus MBPLE for MOFLT Framework. *INCOSE International Symposium*, *33*(1), 601–615. https://doi.org/10.1002/iis2.13041

Madni, A. M., & Sievers, M. (2018). Model-based systems engineering: Motivation, current status, and research opportunities. *Systems Engineering*, *21*(3), 172–190. https://doi.org/10.1002/sys.21438

Matsuo, H. (2015). Implications of the Tohoku earthquake for Toyota′s coordination mechanism: Supply chain disruption of automotive semiconductors. *International Journal of Production Economics*, *161*, 217–227. https://doi.org/10.1016/j.ijpe.2014.07.010

Mocenco, D. (2015). SUPPLY CHAIN FEATURES OF THE AEROSPACE INDUSTRY PARTICULAR CASE AIRBUS AND BOEING. *DOAJ (DOAJ: Directory of Open Access Journals)*.

Paredis Christiaan. (2011). *Model-Based Systems Engineering: A Roadmap for Academic Research*.

pure-systems GmbH. (2024). *Pure Variants. Software for variant management. Version 5.4: pure-systems GmbH*.

R. Karban T. Weilkiens R. Hauber M. Zamparelli R. Diekmann A.M. Hein. (2011). *MBSE Initiative – SE2 Challenge Team: Cookbook for MBSE with SysML*.

Ramos, A. L., Ferreira, J. V., & Barcelo, J. (2012). Model-Based Systems Engineering: An Emerging Approach for Modern Systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(1), 101–111. https://doi.org/10.1109/TSMCC.2011.2106495

Siddique, I. M. (2025). MBSE Implementation in Small Satellite Systems: Rationale for Adoption over Traditional Document-Based Systems Engineering. *International Journal of Geoinformatics Science and Technology*, *1*(1), 1–13. https://doi.org/10.46610/IJGST.2025.v01i01.001

Springer. (2023). Systems Engineering and Product Line Management in the F-35 Program. *Springer*.

Tang, C. S., Zimmerman, J. D., & Nelson, J. I. (2009b). Managing New Product Development and Supply Chain Risks: The Boeing 787 Case. *Supply Chain Forum: An International Journal*, *10*(2), 74–86. https://doi.org/10.1080/16258312.2009.11517219

Tepper, N. A. (2010). *Exploring the use of Model-Based Systems Engineering (MBSE) to develop systems architectures in naval ship design*.

Walden, D. D., Roedler, G. J., & Forsberg, K. (2015). INCOSE Systems Engineering Handbook Version 4: Updating the Reference for Practitioners. *INCOSE International Symposium*, *25*(1), 678–686. https://doi.org/10.1002/j.2334-5837.2015.00089.x

Weilkiens, T. (n.d.). *Systems Engineering with SysML/UML*. http://www.omg.org/.

Weilkiens T., Jesko G. Lamm, Roth S., & Walker M. (2016). *Model-Based Systems Architecture* (Wiley, Ed.).

What was the problem with concordia? (n.d.). 2024. Retrieved March 18, 2025, from https://simpleflying.com/ problems-with-concorde/

Wiley. (2023). Managing Product Lines in Aerospace Engineering. *Wiley*.