AN ONTOLOGY-BASED APPROACH FOR THE CO-DEVELOPMENT AND OPTIMIZATION OF AIRCRAFT CABIN DESIGN AND ASSEMBLY ARCHITECTURES

Yassine Ghanjaoui[1], Jasper H. Bussemaker[1], Jörn Biedermann[1] & Björn Nagel[1]

[1]German Aerospace Center (DLR), Institute of System Architectures in Aeronautics, Hamburg, Germany

Abstract

The aircraft cabin is a highly dynamic environment due to its customizable nature, its short life cycle, its numerous interfaces to aircraft systems, and the continuous integration of new technologies. This impacts the common influence of cabin design and assembly which is highlighted by current industry requirements and applications, such as the need for modularization, system integration and accessibility during installation, or reconfiguration for customized cabin designs and adaptable assembly resources. Therefore, the co-development of cabin designs and assembly planning must be enabled so that both can be assessed and optimized simultaneously. Current methods for the integration of design and assembly focus on abstract business and cost-oriented assessments or consider integration stages where detailed data, such as CAD, CAE, or CAM, already exist. However, there is still a lack of approaches at the conceptual stage that support the technical co-development and optimization of design and assembly. Moreover, there is a need for an automated interpretation of dependencies and interactions between the cabin design and the assembly planning to support the generation of architectural choices for trade-off analyses. This work aims to provide a methodology that integrates conceptual architecture and knowledge models from cabin design and assembly planning to address current research needs. For this purpose, concepts of Model-based Systems Engineering (MBSE) are leveraged, and reusable modeling elements are defined. These depict abstract artifacts of the cabin products and assembly system. The concepts of the two lifecycle stages are semantically mapped and integrated into a consistent Knowledge-Graph (KG). The KG underlies a pre-defined ontology model that links and constrains design and assembly concepts. A specific focus is set on the interfaces between the parts, considering these elements as an intersection point between the product architecture, the emerging assembly processes, as well as the required capabilities and skills of the assembly resources. Furthermore, a generic approach to inferring and translating integrated information from the KG into an architectural design space is developed. This allows for the computation of architecture optimization based on cabin Product-Process-Resources (PPR) variants. It makes it possible to simultaneously plan the assembly processes, allocate the tasks to the resources and enable product architecture trade-offs. This approach supports technical decision-making at an early developmental stage and ensures that traceable, abstract architecture foundlings are available for subsequent detailed analyses.

Keywords: MBSE, Knowledge Integration, Co-Development, DSG, Cabin

1. Introduction

The aeronautical industry is encountering new challenges related to sustainability and the increasing demand for commercial aircraft. Consequently, it must maintain its technological innovation capability by incorporating revolutionary technologies such as hydrogen propulsion or smart systems. The integration of such technologies has a significant impact on the overall aircraft architecture and necessitates early assessments not only in terms of design and construction but also in production and assembly. Furthermore, airlines are seeking more customized, individual designs, leading to

increased manufacturing lead time, cost, and complexity, hindering the possibility of boosting production rates. This challenge is particularly notable for aircraft cabins, which are highly customizable, have short life cycles, and predominantly undergo manual assembly [1, 2]. To assess new, customized cabin designs in terms of production and swiftly plan new assembly processes, close linking and co-development are required. In this context, the concept of a digital thread provides ways to consistently and semantically integrate multidisciplinary modeling and optimization approaches. This End-to-End approach supports the co-development process by enabling data availability and traceability between different cabin design and production artifacts. The integration of detailed product design and production has been addressed by academia and industry, making significant progress with approaches such as Product Lifecycle Management (PLM) [3], digital factory [4], or recent methods such as Systems Lifecycle Management (SysLM) [5] or other similar digital approaches [6, 7, 8, 9]. Many of these approaches focus on business and cost-oriented assessments or consider integration stages where detailed data (CAD/CAE/CAM) already exists, restricting solution space exploration by design decisions already made. This limits, on one side, the assessment of all feasible product and production designs and, on the other side, constrains the development of production processes and resources to specific product configurations. However, integration should be enabled at earlier conceptual stages where abstract interdependencies are identified and leveraged for co-development. Recent research approaches [10, 11, 12, 13] that focus on early integration at the conceptual stage recognize the importance of leveraging Model-based Systems Engineering (MBSE) to support modeling interdependencies between product and production systems in the interdisciplinary conceptual stage. MBSE is an approach that uses formalized representations in the form of models to support activities related to the design, analysis, verification, and validation of systems throughout their lifecycle. The modeling approaches deliver holistic ways for the analysis and representation of the two domains. However, there is a specific lack of focus on the causal relationships between model elements and how these can be used to specify and constrain the design space for products and resources, as well as generate assembly processes. Moreover, many of the presented approaches in the literature also limit themselves to manual exploration and implementation, hindering the automation of the co-development and optimization process.

This research presents an approach that aims to support cabin design and assembly co-development and optimization at early conceptual stages. Through the definition of specific modeling elements in the design and assembly architectures, both domain are semantically integrated. The suggested approach defines an ontology that depicts the relationships between the two domains and supports the semantic knowledge integration and reuse in a tool-agnostic way. The application case that is shown in this work demonstrates how integrated knowledge can be converted into design spaces, enabling automatic exploration of cabin product design and assembly.

2. Methodology for Co-Development and Optimization of Cabin Architectures

In this work, a methodology has been developed to capture and integrate abstract information about cabin design and assembly into a common design space at an early development stage. Figure 1 illustrates the proposed methodology for the integration and optimization of abstract cabin design and assembly architecture models. In conventional approaches, system design is started from requirement and proceeded by developers from different disciplines through decision making based on their knowledge and expertise. Instead, the presented methodology integrates multidisciplinary abstract models and transforms them into an optimization problem where architectures are generated, evaluated and optimized automatically. Best architecture instances are fed back to respective disciplines models, supporting a value driven development.

The methodology is based on three main components. The first component represents the model-based architectures, where relevant artifacts for both cabin design and assembly systems are defined and modeled. These represent relevant design space elements for the optimization problem. The second component deals with the integration of information from these two domains into a single knowledge graph. This involves creating a formal ontology description of the multidisciplinary concepts, which is then used to map information from the architecture models. Finally, the third component demonstrates how formal knowledge is converted into a Design Space Graph (DSG). This graph can be leveraged for the evaluation and optimization of potential multidisciplinary architecture

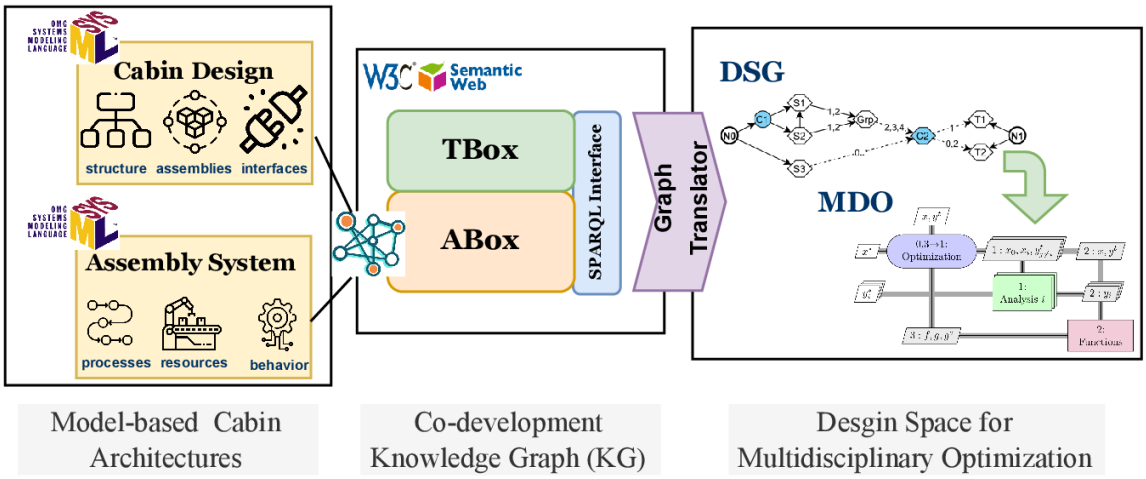


Figure 1 – Methodology for integration and optimization of abstract cabin design and assembly architecture models

instances. Each of these components is described in the following sections.

2.1 Model-based Cabin Architectures

To enable an early abstract representation of a cabin's design and assembly, this work proposes a methodology that leverages Model-Based Systems Engineering (MBSE) to integrate the architectures of both disciplines into a multidisciplinary design space. MBSE is the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities, beginning in the conceptual design phase and continuing throughout development and later life cycle phases [14]. The Systems Modeling Language (SysML) is widely adopted in the MBSE context and provides notations, syntax, and semantics to support architectural modeling activities [15].

The literature provides extensive methodologies to leverage SysML for generic system development [16, 17, 18, 19] as well as specifically for cabin architectures [20, 21, 22]. Many of these methodologies follow a top-down approach, starting from a black-box system analysis where stakeholder needs are identified and the system context is analyzed. The analysis is conducted on different abstraction layers of the system—functional, logical, and physical—leading to architectural decisions at each level. The resulting model artifacts represent a conceptual description that is then available and reusable for further architecting and analysis activities.

The first methodological step is to define specific elements that extend the cabin design model in terms of assembly factors. These elements provide the means to represent potential architectures, identify architectural decisions, and define performance metrics for evaluating different architectures. They are specified based on the interdependencies between design and assembly. Most of the modeling artifacts are typically generated at a logical or physical abstraction layer, as outlined in the referenced methodologies. They can be grouped into three categories: the cabin structure, the cabin composing assemblies, and the components' interfaces for assembly.

The generic definition of the modules assembled in the cabin can be represented in a block definition diagram. With multiplicity specification, it is possible to define which modules are mandatory parts of any cabin instance and which are optional. For instances that do not include these architectural options, all related elements, such as interfaces, are automatically removed. The cabin decomposition is realized in terms of hierarchical subdivisions (e.g. modules, components, parts). Each cabin element has a set of properties that distinguish different variant implementations. By redefining these properties, specific variant values can be assigned to the properties.

From an assembly perspective, the way cabin elements are interconnected plays a major role. The interconnection is typically depicted in an MBSE approach through the connected system parts via proxy ports. The generic interconnection can still be used to instantiate different architectures by selecting or deselecting the parts and ports. The approach extends the representation by defining

an element `Assembly Interface` as a specialization of the SysML element `Interface Block`, which is used to type the relevant ports for the assembly process. These interfaces define the problem space, i.e., the requirements based on the two assembled elements and the flow between them. They do not specify the joining technology to be used as a solution, as this is one of the intended results of the ontology-based evaluation and optimization.
The Assembly Interfaces need to be linked to different types of requirements (functional, non-functional, mechanical, electrical, integration, test) and are reusable at different interfaces in the architecture. The mentioned requirements have also been specified with new stereotypes and all have the element `Interface Requirements` as a generalization. Here, the requirement can be described and related properties such as load or pressure are defined. The required values can be given in concrete or abstract ranges, as exact values are not easy to determine at this development stage.
In the next step, relevant model artifacts from the assembly system architecture are defined for later integration. Both "greenfield" and "brownfield" assembly planning scenarios are considered. Brownfield refers to the reuse and reorganization of existing assets to meet new or adjusted production requirements while considering their limitations. In contrast, greenfield production systems start from scratch, allowing for the optimal design of assets according to current best practices [23].
In this work, the brownfield approach is initially adopted, considering that production systems are often built upon previous ones for new aeronautics products. The capabilities provided by one or multiple combined resources can be derived from the functional analysis of the available resources. The skills of the resources, i.e., how the capabilities are implemented by different resources, are derived from the logical or technical resource analysis.
The greenfield approach is applied to the assembly processes that require capabilities not provided by existing resources. These capabilities can either be assigned to specific resources by extending their functional behavior or by designing new resources with the required functionality.

2.2 Co-Development Knowledge Graph

To capture different domain concepts, ontologies have shown potential for semantic knowledge integration and reuse in a tool-agnostic way [24]. According to [25], an ontology defines "a vocabulary of concepts and some specification of their meaning. This includes definitions and an indication of how concepts are interrelated, which collectively impose a structure on the domain and constrain the possible interpretations of concepts." The definition of the concepts, their attributes, and the relations between them is realized in the so-called Terminology Box (TBox), while the definition of asserted individuals and instances is part of the Assertion Box (ABox) [26, 24].
The Knowledge Graph (KG) consists of a TBox and ABox and can be used to retrieve (query) information that is either explicitly defined or implicitly inferred through reasoning [27]. KGs can be serialized in the Web Ontology Language (OWL), a W3C standard that allows the specification of formal semantics. OWL is based on the Resource Description Framework (RDF), which describes information as RDF triples, where each triple is a subject–predicate–object statement (e.g., Cabin-has Module-Galley), which can be queried using the SPARQL Protocol and RDF Query Language (SPARQL) that uses the same notation [28].
To create a KG, data from engineering tools can be directly imported into the KG (a process known as materialization), or KGs can access data from different engineering artifacts, a method called ontology-based data access (OBDA) [27]. Additionally, OWL can represent description logic and contains a sequence of axioms and facts that enable reasoning. OWL can also incorporate Semantic Web Rule Language (SWRL) rules to provide a more complete ontology and enable rule-based reasoning [29]. For example, a simple rule could assert that the combination of the `hasParent` and `hasBrother` properties implies the `hasUncle` property.
Especially in the manufacturing field, significant progress has been made in consistently abstracting domain information. Formalized guidelines and related ontologies are available and provide expert knowledge representation that maps concepts related to products, processes, and resources [24, 30, 31, 32, 33, 34]. These ontologies are used and extended in this work to integrate the model artifacts presented in section 2.1 The ontology is defined using OWL to semantically link the main artifacts from the cabin design and assembly system models. Figure 2 shows a lightweight representation of the TBox conceptualization as a UML class diagram.

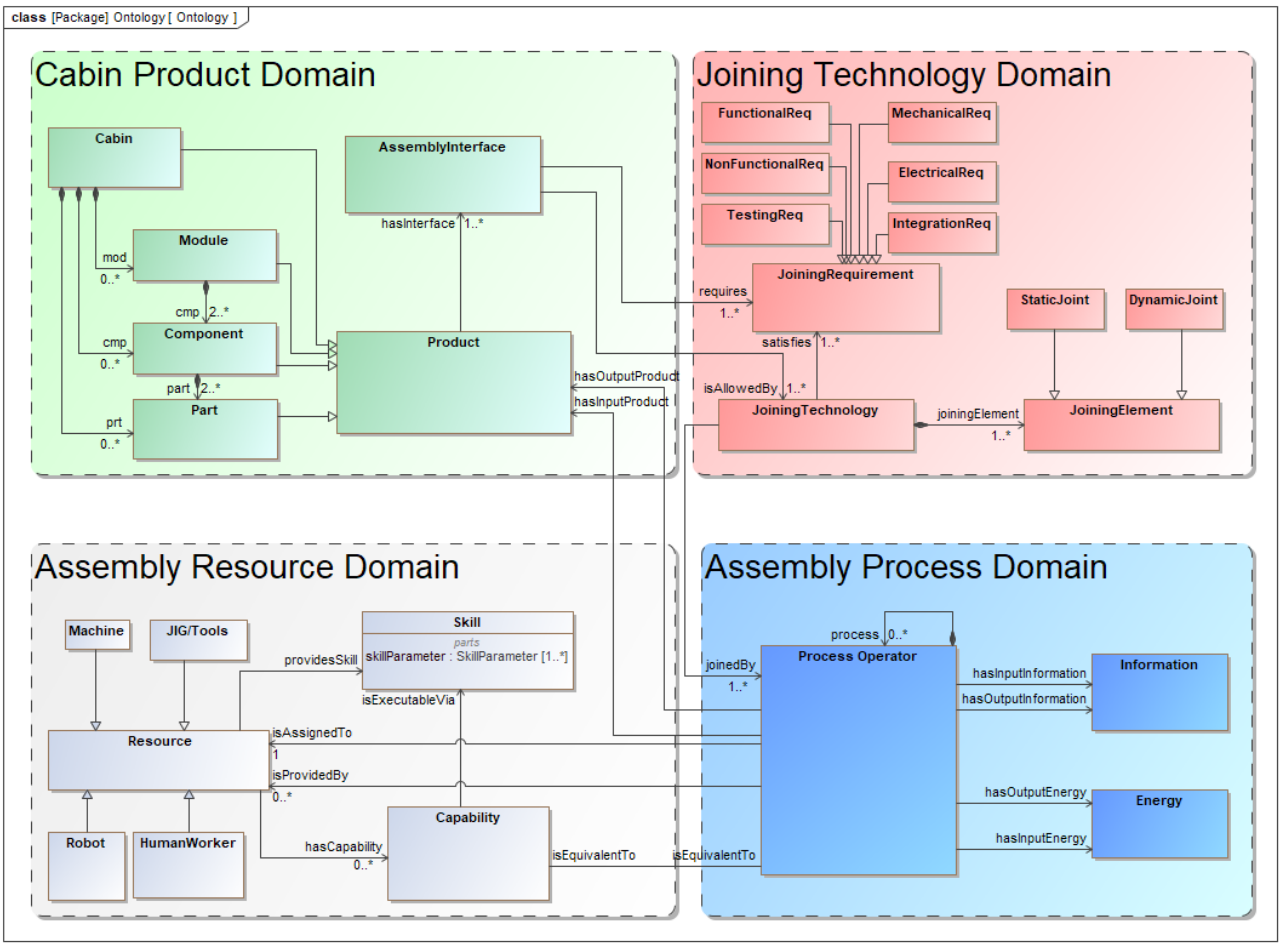


Figure 2 – TBox of cabin design and assembly ontology

The presented TBox can be subdivided into four interlinked domains. The cabin product domain includes the model artifacts that emerge from the conceptual design analysis. These artifacts are linked to the joining technology domain through the interface requirements for the existing cabin products. In this domain, the joining technology elements are related to these requirements based on satisfaction constraints. The `JoiningTechnology` concept can be defined as a technical solution library with specifications of data properties related to the different requirement categories.
Both mentioned domains are linked to the assembly process domain, where processes are described (as defined in the VDI 3682 guideline [24]) as a transformation of energy, information, and products. As part of the joining technology specification, the required processes to implement it are also defined, hence the semantic link in the ontology. Finally, the assembly resource domain integrates knowledge emerging from the brownfield artifacts from the assembly model. The resource capabilities are set, as described in [33], as semantically equivalent to the processes, which enables the assembly process allocation. The skills and skill parameters allow differentiation between the resources in terms of a common capability executed in different manners.

2.3 Design Space for Optimization Problem Solving

Applying multidisciplinary optimization enables the consideration of various architecture combinations with different multidisciplinary parameters and allows for selective evaluation, thus saving time and computational resources. Especially in early development stages, where the variety of concepts is very large, this supports value-driven decision-making that is not tied to one specific domain or solely based on expert knowledge. In this context, the Design Space Graph (DSG) can be used to formulate the optimization problem in terms of variables, objectives, and constraints, which can be utilized by optimization algorithms to explore the design space [35, 36].
A DSG is a directed graph that combines architecture elements (e.g., functions and components) with

decision models, enabling the automatic generation of architecture candidates. The DSG consists of two domains: a selection domain with generic nodes, derivation edges, incompatibility edges, and selection choices for selecting the elements included in an architecture instance; and a connection domain with connector nodes, connection edges, and connection choices for modeling connection tasks. Moreover, it is possible to define generic design variables, e.g., to model parameter selection. These are defined using design variable nodes for continuous and discrete design variables.
The design problem definition also requires the definition of performance metrics using metric nodes. These represent the output of the evaluation function and are used as objectives or constraints in the optimization problem. Therefore, the evaluation function must be defined to return the performance for a given DSG instance in terms of the defined output metrics. The optimizer runs the evaluation on the design space iteratively to find the optimum.
In this work, a graph translator has been developed to access formalized knowledge in the KG and utilize semantic information to formulate the optimization problem using the DSG. The translator includes an API containing SPARQL queries that return asserted information for different types of elements or relationships. The information is processed, and a DSG is automatically built using the DSG Core API [37].
First, the translator begins by creating the selection choices. It queries specific object properties (e.g., `isComposedOf`, `joinedBy`) in the KG and processes their constraints and multiplicity to create selection nodes and the corresponding selection possibilities as further nodes. The translator then queries the data properties in the graph to create metrics in the DSG for the created nodes. In the next step, specific rules defined with SWRL are utilized to create links between different nodes. The translator uses reasoning results and creates connection nodes between the corresponding nodes.
Furthermore, other logical constraints such as disjunctiveness are used to reduce the number of possible architectures by creating incompatibility edges in the DSG. These edges assert that if either of the two nodes is confirmed, the other node and its derived nodes are not. Post-processing activities are then required to set initial nodes that are relevant to generate valid architectures and also to define which metrics are to be used as objectives.
After a DSG is defined, the optimization problem as shown in Figure 3 can be run. The DSG is encoded into design variables that are accessed by the optimization algorithm every iteration.

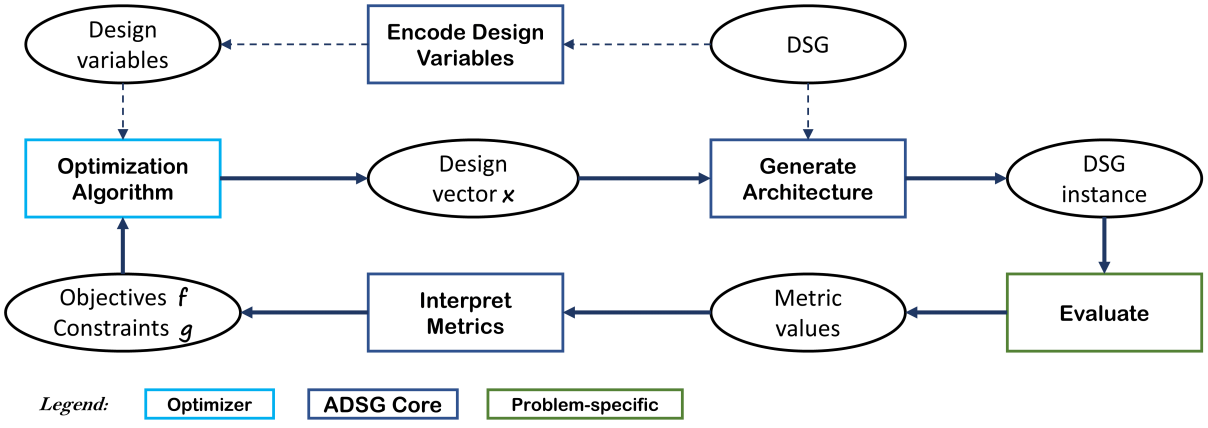


Figure 3 – Optimization workflow using DSG (modified from [37])

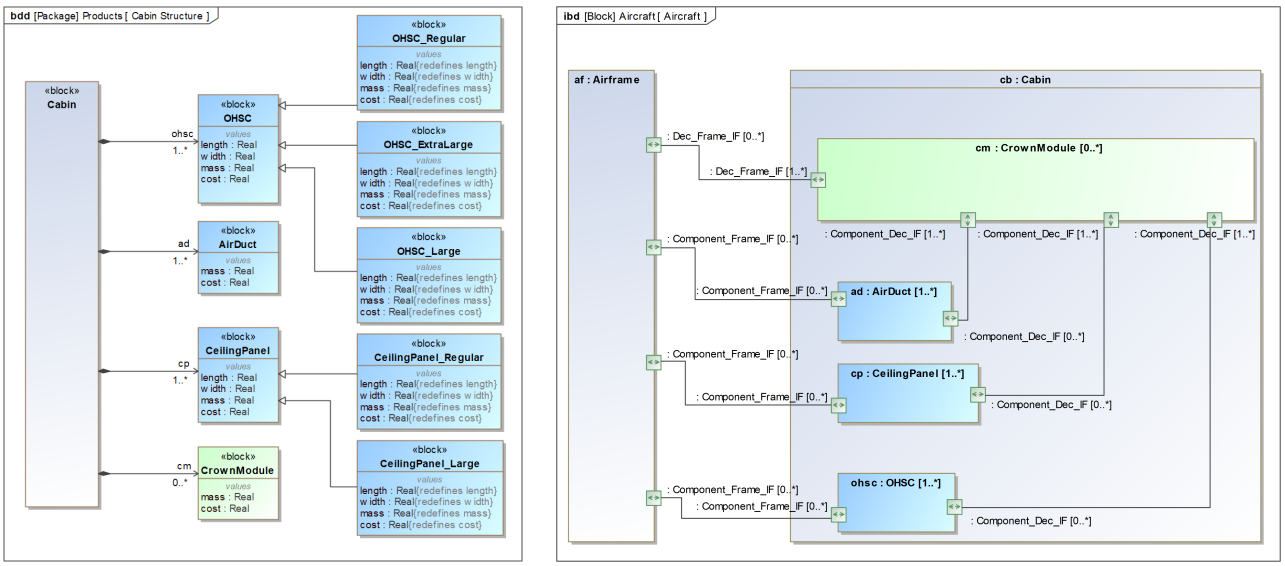
A selection of a corresponding algorithm can be made based of the nature of the DSG and the intended results. The optimization algorithm suggests a design vector, that is decoded as a DSG instance. The latter is evaluated according to the problem-specific evaluation function. It generates metrics values that are interpreted as objectives and constraints. These are communicated back to the algorithm for next iterations until optimal solutions are found.
The optimization problem is run by coupling to SBArchOpt, which an open-source library for running architecture optimization problem [38]. The python library *ADSG Core* provides a problem definition in the API of SBArchOpt, so that all algorithms defined in SBArchOpt can be used.

3. Application Case: Co-Development and Optimization of Decoupled Cabin Architecture

In this section, a use case from the cabin context serves as a proof-of-concept to demonstrate the applicability of the methodology shown in section 2. This use case involves decoupling the cabin from the airframe, an approach that is gaining importance in aircraft manufacturing research. The benefits of decoupling include simplified installation processes, reduced reconfiguration costs, and shorter lead times. One technical solution is the Crown-Module (CM), a frame structure that decouples the ceiling and hat-rack areas, where electrical components, air conditioning, oxygen supply, and other mechanical systems are located. Although integrating a CM offers many benefits, it can also introduce design and assembly challenges. Therefore, it is crucial to conduct co-development and multidisciplinary optimization of both coupled and decoupled architecture instances at an abstract level. This approach helps understand the overall impact of this technical solution before proceeding with detailed geometrical and industrial design. In the following sections, the developed methodology is applied to these architectures, demonstrating how the application and integration of MBSE, ontologies, and design spaces can support early synergy assessment.

3.1 Model-based Architectures of Cabin and Assembly System

The modeling activities begin by defining the structure of the cabin, including both coupled and decoupled variants. The block definition diagram in Figure 4a illustrates this structure. By specifying multiplicity, it is possible to assert that every cabin instance is composed of modules placed in the ceiling area, and that some cabin instances include a CM. Using the generalization relationship, different variants of the Overhead Storage Compartments (OHSC) and ceiling panels can be defined. These variants have different design properties in terms of geometry, mass, and cost, which are used during the evaluation for the overall assessment.



(a) Configurable cabin structure

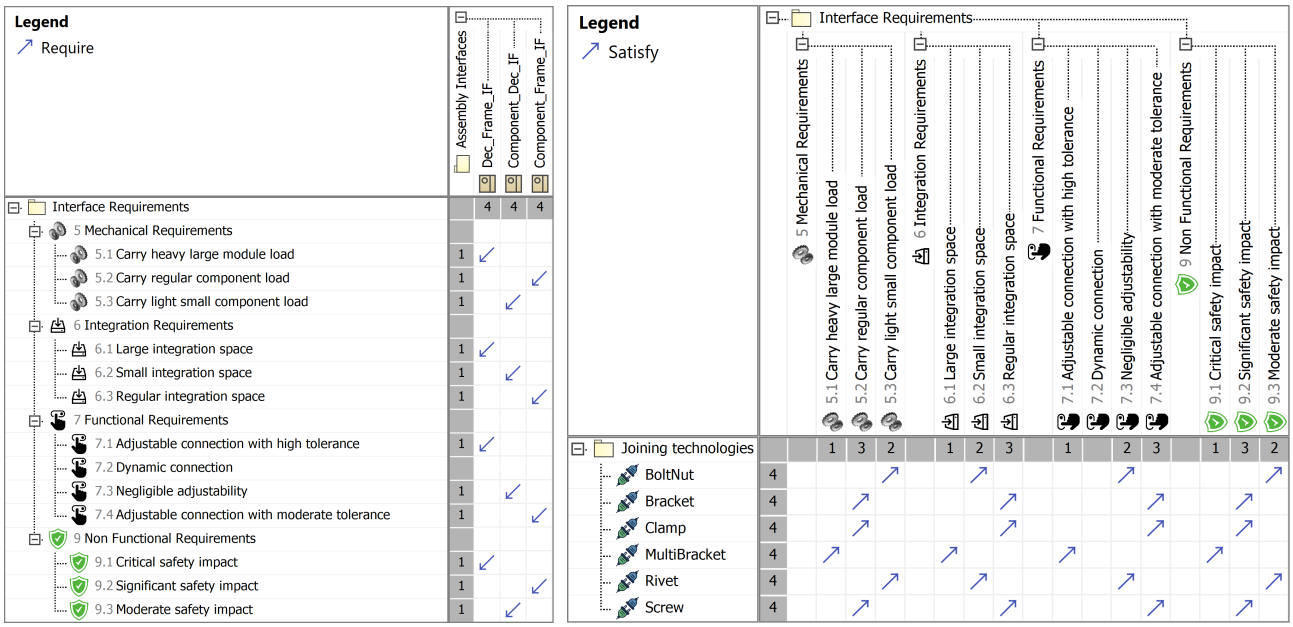
(b) Cabin's internal interfaces

Figure 4 – MBSE architecture for cabin decoupling

Furthermore, the internal structure, including the interfaces between the cabin modules, is depicted using an internal block diagram as shown in Figure 4b. Three different types of interfaces have been defined using Interface Blocks: interfaces between cabin modules and the frame, between cabin modules and the CM, and between the frame and the CM. This classification is due to the common requirements for these interface types. The proxy ports of the parts are typed by these Interface Blocks using the corresponding multiplicity, which is used to generate interfaces for both decoupled and coupled instances.

To model the link between the assembly interfaces and their corresponding requirements, a stereotype `Require` has been defined. Additionally, new requirement stereotypes have been introduce

for different types of requirements. Figure 5a shows the matching results between the assembly interfaces and requirements. These are defined as abstract requirements regarding the mechanical load to be carried, the required space for integration and installation, the expected functionality of the interface technology, as well as safety-related requirements. Moreover, a library of joint technologies has been defined and linked to the specified requirements using the SysML relationship `Satisfy`. Figure 5b shows the results of this matching. Both of these relationships are used when integrating model information into the KG according to the ontology depicted in Figure 2.



(a) Interface requirement matrix (b) Satisfaction matrix for joining technologies

Figure 5 – Interface matching according to ontology

The second step in the modeling process pertains to the assembly system. As mentioned in section 2.1, a brownfield approach is adopted by modeling available resources. In this use case, resources at the laboratory of the Institute of System Architectures in Aeronautics (DLR-SL) have been considered. Using the semantic definition, properties regarding their capabilities and skills have been defined through the instantiation of the resources block definition. Figure 6 shows the resulting instance for the assembly system. The system is composed of a worker, tooling, as well as two automated resources: a static robot and a mobile robot. The corresponding capabilities (see purple blocks in Figure 6) are shown, and the related skills with respective parameters (see red circles in Figure 6) for each resource and capability are specified. The instance specification values are used for integration into the KG.

Since processes and their allocation to the resources are one of the results of the intended optimization, they are integrated at a later development stage, and the results are shown in section 4.

.

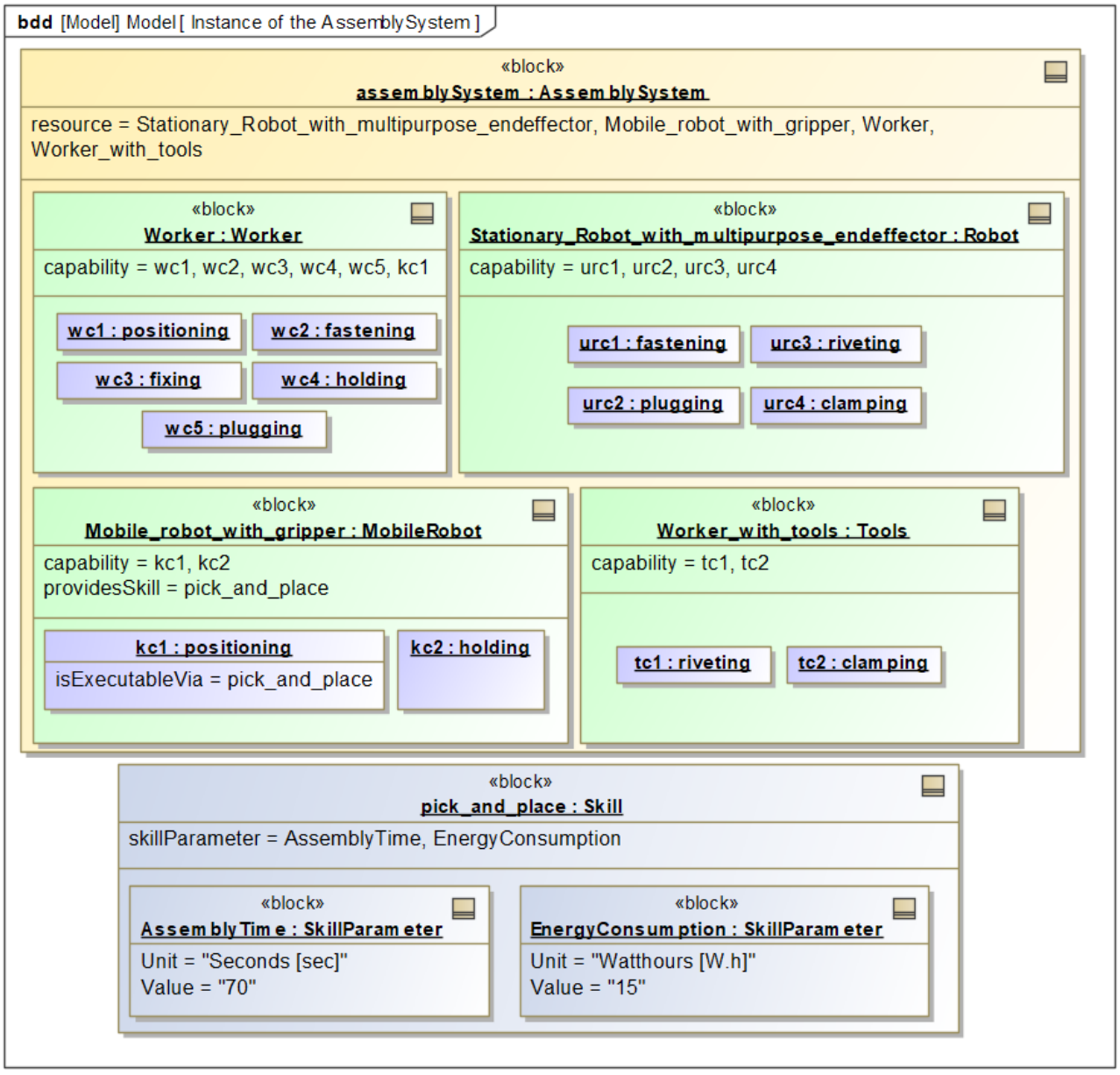


Figure 6 – Instance specification of the assembly system with available resources

3.2 Integration of Cabin Design and Assembly Knowledge in KG

The information captured in the SysML Models for design and assembly are integrated into the KG by parsing the models using the XML Metadata Interchange (XMI), which is an interchange format for SysML metadata [39]. The stereotypes are used to map specific model elements to the TBox-concepts and collect corresponding information. The latter are then used to create the ABox with all individuals and relationships. To results are visualized using the open-source ontology editor *Protégé* and are shown in Figure 7.

The KG illustrates the two cabin variants and their composing modules. The modules are linked to their possible interfaces, which require specific interfacing requirements as defined in Figure 5a. The joining technologies are also linked to these requirements based on the satisfaction constraints defined in Figure 5b. Each of these joining technologies implies the necessary assembly processes, which are aligned with the capabilities of available resources.

Furthermore, two rules defined in SWRL are added to the TBox semantics and are used for the assertion of individuals in the ABox. The SWRL notation are defined as follows:

$$\text{RL1:}\quad \text{requires}(?int, ?req) \wedge \text{satisfies}(?tech, ?req) \rightarrow \text{isAllowedBy}(?int, ?tech) \tag{1}$$

$$\text{RL2:}\quad \text{hasCapability}(?res, ?cap) \wedge \text{isEquivalentTo}(?cap, ?proc) \rightarrow \text{isProvidedBy}(?proc, ?res) \tag{2}$$

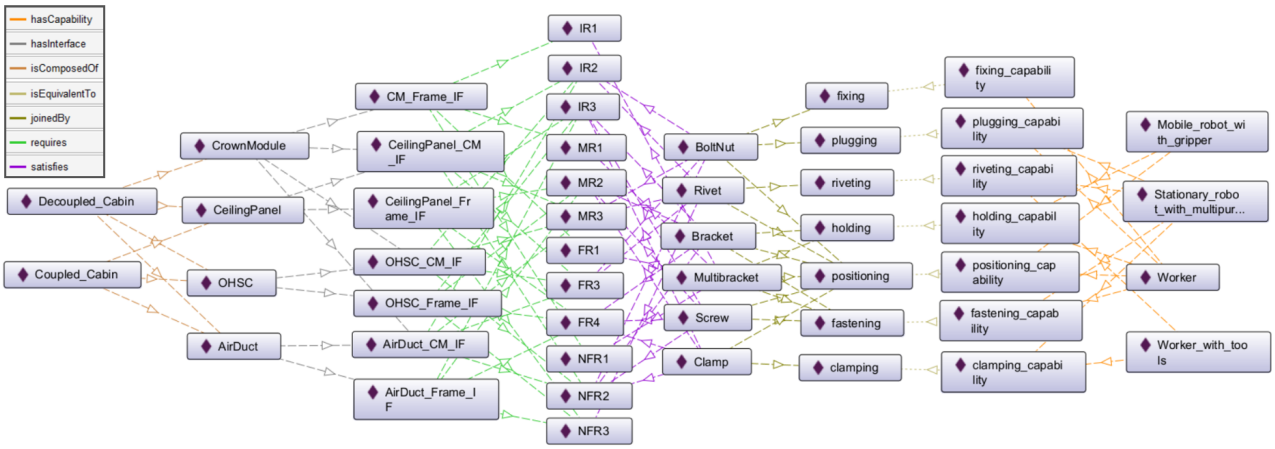


Figure 7 – Resulting ABox for linked data from SysML models of cabin design and assembly

The first rule sets the compliance of a module interface with a joining technology, if the latter satisfies all requirements needed for a specific interface. The second rule asserts that a process can be provided by a resource, if the latter has the required capability associated to this process. These rules are used by the inference engine and generate additional relationships, that are used by the DSG translater to build the design space.
Moreover, the translator also utilitized SPARQL queries to parse information from the ABox. These are defined as follows:

```
Q1:
PREFIX co:
<http://www.dlr.org/onto/cabin#>
SELECT ?cabin ?Module ?Interface
WHERE {
?cabin a co:Cabin.
?cabin co:isComposedOf ?Module.
?Module co:hasInterface ?Interface
}
```

```
Q2:
PREFIX co:
<http://www.dlr.org/onto/cabin#>
SELECT ?Joint ?Process
WHERE {
?Joint a co:JoiningTechnology.
?Joint co:isjoinedBy ?Process.
}
```

The URI-link defined for the prefix "`co`" is pointing at TBox concepts and relationship in the developed cabin ontology. The first query is used by the translator to get all composing modules for each cabin architecture, and their corresponding interfaces. The requested information is retrieved using the "`SELECT`" query form, which return variables bound in a query pattern match. The second query retrieves processes that are needed for the assembly using specific joining technologies.

3.3 Evaluation and Optimization of Cabin Architecture

To analyze the architecture's impact on cabin design and assembly, two optimization objectives have been defined for each domain. The design is evaluated based on total product mass and costs, while the assembly is evaluated based on time and energy consumption. An evaluation function for each of these metrics is defined. The evaluation function retrieves information regarding assembly processes, masses, and costs from an evaluation database. This database contains assumption-based information about each module type in combination with available joining technologies. The evaluation function uses this data to calculate the number of joining elements and assembly processes required for an architecture instance. It computes the mass and costs of a design by summing the masses and costs of the modules and joining elements. Similarly, it computes the assembly time and energy consumption by summing the individual data for all processes required for a specific architecture instance. In this work, no process sequencing is implemented in the evaluation. Therefore, the computed assembly time does not consider possible overlapping of parallel processes or the temporal availability of resources during process execution.
The optimization is subsequently run using the Non-dominated Sorted Genetic Algorithm II (NSGA-

II), a sorting-based multi-objective evolutionary algorithm [40]. This algorithm is well-suited for large design spaces with an exhaustive list of candidate solutions. It calculates the Pareto front, corresponding to a set of optimal, nondominated solutions. The algorithm maintains a set of individuals, collectively referred to as a population. Each individual represents a solution. During evolution, the population is updated in each generation, and new individuals are created. The evolutionary process ends when the population approaches the Pareto front.

4. Results and Discussion of Architecture Design Space and Optimization

In this section, main results for the architecture design space and optimization for the cabin decoupling application case are presented. By running the translator, the DSG selection and connection nodes, connection and derivation edges as well as metrics are provided. Additional post-processing activities are implemented to select initial and metric nodes. Figure 8 shows the results of the design space and the graph elements parsed using the queries *Q1* and *Q2* are highlighted.

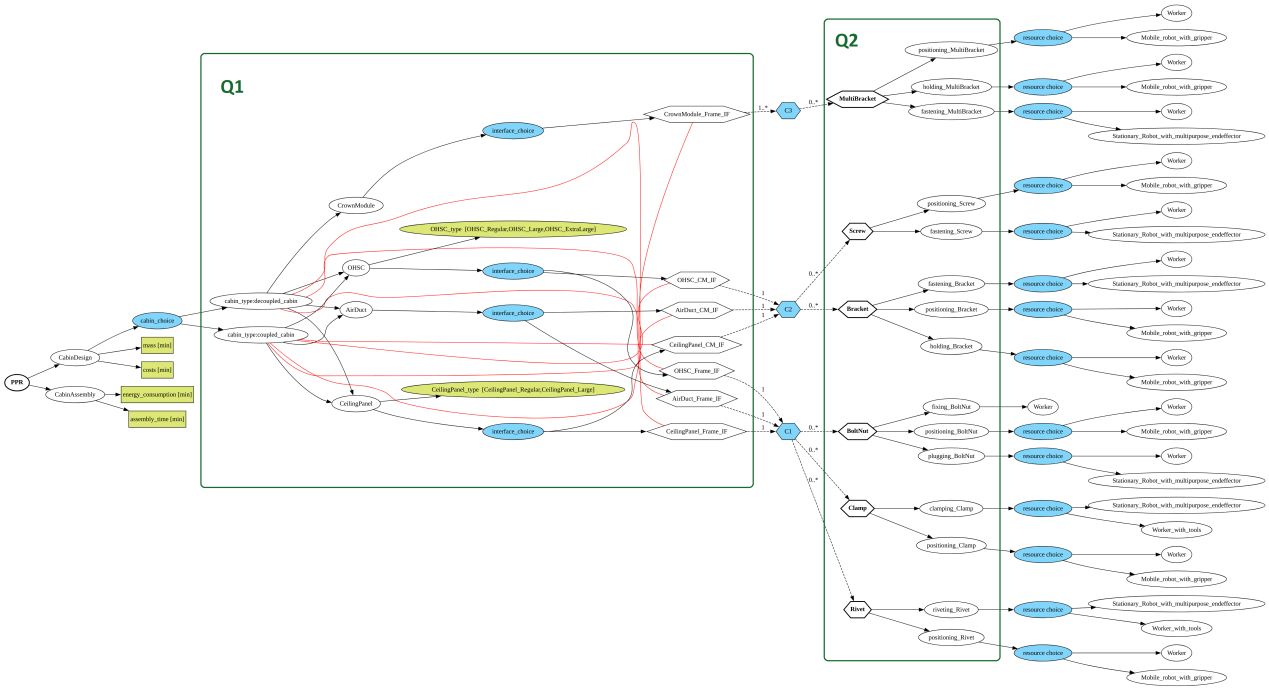


Figure 8 – Resulting DSG for coupled and decoupled cabin architectures

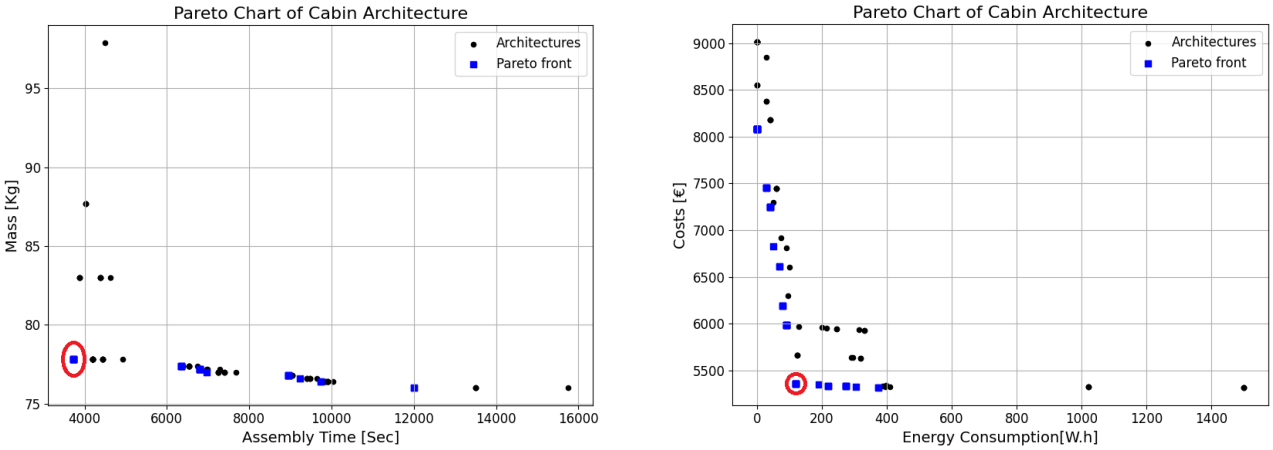
The selection of decoupled or coupled cabin architecture determines whether the CM is included or not. The OHSC and ceiling panels include different variants and are specified with the design variable nodes. As defined in the SysML model, each cabin module can have a direct interface to the airframe or be assembled via the CM. The interfaces are specified as selection choices and are only available for the corresponding architecture through the incompatibility edges (see red lines). The interface nodes are connected to the joining technology nodes using inferred information in the ABox by applying rule *RL1*. The connection edges, shown by dashed black arrows, display the connection constraints regarding the allowed number of incoming and outgoing connections. Moreover, the assembly processes required for each technology are associated with plausible resources using selection choices. These are defined based on the logic in rule *RL2*.
The optimization is subsequently run for the DSG. Parameter regarding the size of the design space and the optimization computation time are shown in Table 1.

Table 1 – Properties of design space optimization

Valid design space	Number of objectives	Population size	Number of generations	Computation time for problem optimization (sec)
3,440,640	2 for each problem definition	100	10	41.85

In this work, two optimization cases are shown where two objectives are selected. The first case opti-

mizes the cabin's products mass an the assembly time, and the second case optimizes the cabin products costs and the energy consumption during assembly. The results of the multi-objective pareto-optimization are shown in Figure 9.



(a) Pareto chart for mass and assembly time optimization

(b) Pareto chart for costs and energy consumption optimization

Figure 9 – Multi-objective optimization results for cabin design and assembly using NSGA-II algorithm

In the Pareto charts, population individuals are represented as black dots, and optimal Pareto front individuals are represented by blue squares. Each of these individuals represents a cabin architecture instance. In both cases, no Pareto optimal result is available, but the best architectures are selected according to architectural preferences and highlighted with red circles in Figures 9a and 9b. In the first case, the selected optimal architecture has an assembly time of **61.8 minutes** and a product mass of **77.82 kg**, and is shown in Figure 10. In the second case, the selected optimal architecture has an energy consumption of **118 Wh** and product costs of **5362€**, and is shown in Figure 11.

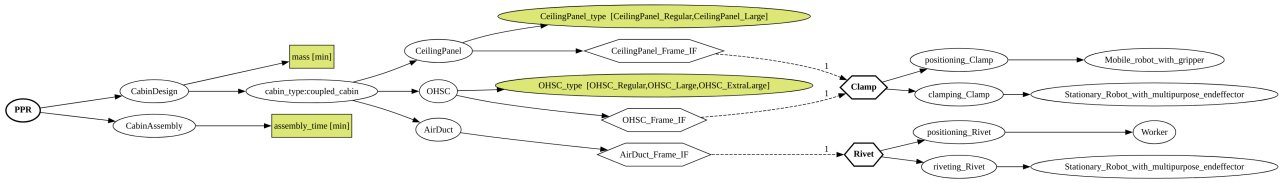


Figure 10 – Cabin architecture instance with best mass and assembly time results

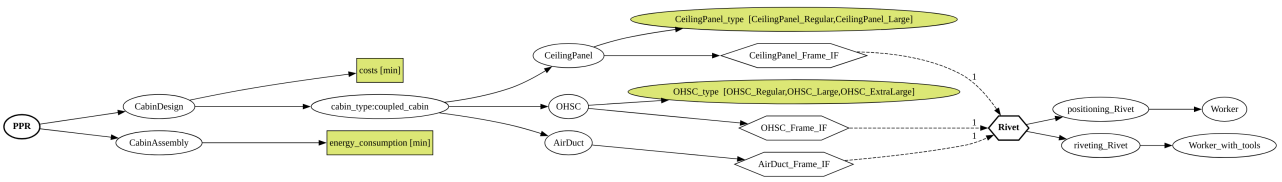


Figure 11 – Cabin architecture instance with best costs and energy consumption results

Both instances involve a decoupled architecture without a CM. The additional mass, costs, assembly time, and energy consumption of this module are not mitigated by lighter and reduced joining technologies. Consequently, only bolts with nuts, rivets, and clamps are available for directly assembling the modules to the frame. In the optimal instance of the first case, clamps are used for the assembly of the ceiling panel and OHSC, while rivets are used for the air-duct assembly. The additional process required for bolts with nuts adds more time, which is why this joining technology wasn't chosen. Furthermore, mostly automated resources are assigned to these processes due to their faster execution

for these specific types. In the optimal instance selected for the second case, rivets are used for all modules. Apart from their favorable costs, the required processes can be managed by workers and corresponding tools with very low energy consumption.
The methodology presented in this work leverages and combines various technologies. MBSE enables the depiction of design artifacts and domains independently in two abstract models. Semantic web technologies such as RDF, OWL, and SPARQL are used for building and processing the ontology and knowledge graph. These serve as the integration core, where interactions and dependencies among different domains are represented. Thus, domain experts can develop their domain-specific MBSE models, and ontology experts can integrate their results into the common KG. Finally, design space exploration and optimization technologies, such as DSG and optimization tools, utilize the integrated knowledge to find optimal solutions and enable multidisciplinary trade-off analyses. A translator between the knowledge graph and DSG links the KG with optimization activities and facilitates the assessment different cabin architecture instances.
The application case demonstrates how multidisciplinary cabin architectures can be assessed in early development stages. A broad design space incorporating numerous architectural choices can be evaluated efficiently, facilitating architecture optimization and data-driven trade-offs. However, the evaluation data used in this study are based on assumptions and should be validated through high-fidelity simulations or hardware execution. Additionally, aspects that are not adequately captured in the ontology definition or the evaluation, such as very abstract requirements descriptions or the omission of process sequencing, can lead to unrealistic assessments.

5. Conclusion and Outlook

To tackle challenges related to aircraft cabin customization and manufacturability, evaluating diverse multidisciplinary architectures early in the design process is crucial. This evaluation should encompass the overall impact of cabins on design and assembly. This work introduces an ontology-based approach for co-development and optimization in the aircraft domain. Leveraging Model-Based Systems Engineering (MBSE), descriptive abstract models depicting design and assembly artifacts are independently developed. These models are integrated into a unified Knowledge Graph (KG) that defines a cross-domain ontology, facilitating the assertion of individuals through inferred knowledge.
The cabin application case presented demonstrates how the KG can be practically translated into design spaces and utilized for multidisciplinary optimization. Deriving design space from semantic knowledge representation ensures consistency, reusability, and scalability for detailed analyses. Thus, the application case validates the feasibility of optimizing large cabin design and assembly spaces at an abstract development stage.
However, the evaluation data in this work are based on assumptions, posing limitations on the plausibility of assessment results. Future research should focus on developing detailed models of products, joining technologies, and assembly resources to generate more realistic data. These models could stem from detailed simulations or real hardware, accounting for uncertainties typical of early development stages. Therefore, extending the ontology representation to encompass complex concepts and relationships that align with detailed simulation and hardware models is essential.
Finally, future work will explore quantitative comparisons between experimental results and those generated by abstract assessments to assess the reliability of the co-development approach.

6. Contact Author Email Address

mailto: yassine.ghanjaoui@dlr.de

7. Copyright Statement

References

[1] Fabian Laukotka, Jan Oltmann, Dieter Krause, et al. A digitized approach to reduce assembly conflicts during aircraft cabin conversions. In *DS 98: Proceedings of the 30th Symposium Design for X (DFX 2019)*, pages 251–262, 2019.

[2] Kiara Ottogalli, Daniel Rosquete, Javier Rojo, Aiert Amundarain, José María Rodríguez, and Diego Borro. Framework for the simulation of an aircraft final assembly line. In *MATEC Web of Conferences*, volume 233, page 00010. EDP Sciences, 2018.

[3] Yasushi Umeda, Shozo Takata, Fumihiko Kimura, Tetsuo Tomiyama, John W Sutherland, Sami Kara, Christoph Herrmann, and Joost R Duflou. Toward integrated product and process life cycle planning—an environmental perspective. *CIRP annals*, 61(2):681–702, 2012.

[4] H Bley and C Franke. Integration of product design and assembly planning in the digital factory. *CIRP Annals*, 53(1):25–30, 2004.

[5] Martin Eigner and Mona Tafvizi Zavareh. Digitalization of the engineering supported by system lifecycle management (syslm). In *IFIP International Conference on Product Lifecycle Management*, pages 240–254. Springer, 2021.

[6] Giuseppa Donelli, Pier Davide Ciampa, João MG Mello, Felipe IK Odaguil, Ana PC Cuco, and Ton van der Laan. A value-driven concurrent approach for aircraft design-manufacturing-supply chain. *Production & Manufacturing Research*, 11(1):2279709, 2023.

[7] Louis Schäfer, Antonia Frank, Marvin Carl May, and Gisela Lanza. Automated derivation of optimal production sequences from product data. *Procedia CIRP*, 107:469–474, 2022.

[8] Masoud Rais-Rohani. Manufacturing and cost consideration in multi-disciplinary aircraft design. *NASA Grant NAG-1-1716, NASA Langley*, 1996.

[9] Xiaochen Zheng, Xiaodu Hu, Rebeca Arista, Jinzhi Lu, Jyri Sorvari, Joachim Lentes, Fernando Ubis, and Dimitris Kiritsis. A semantic-driven tradespace framework to accelerate aircraft manufacturing system design. *Journal of Intelligent Manufacturing*, 35(1):175–198, 2024.

[10] Markus Christian Berschik, Marvin Blecken, Hiteshkumar Kumawat, Jan-Erik Rath, Dieter Krause, Ralf God, and Thorsten Schüppstuhl. A holistic aircraft cabin metamodel as an approach towards an interconnected digitised cabin lifecycle. In *33rd Congress of the International Council of the Aeronautical Sciences, ICAS 2022*. The International Council of the Aeronautical Sciences, 2022.

[11] Louis Schäfer, Matthias Günther, Alex Martin, Mariella Lüpfert, Constantin Mandel, Simon Rapp, Gisela Lanza, Harald Anacker, Albert Albers, and Daniel Köchling. Systematics for an integrative modelling of product and production system. *Procedia CIRP*, 118:104–109, 2023.

[12] Chantal Sinnwell, Nicole Krenkel, and Jan C Aurich. Conceptual manufacturing system design based on early product information. *CIRP Annals*, 68(1):121–124, 2019.

[13] Albert Albers, Tobias Stürmlinger, Constantin Mandel, Jiaying Wang, Marta Bañeres de Frutos, and Matthias Behrendt. Identification of potentials in the context of design for industry 4.0 and modelling of interdependencies between product and production processes. *Procedia CIRP*, 84:100–105, 2019.

[14] Jonas Hallqvist and Jonas Larsson. Introducing mbse by using systems engineering principles. In *INCOSE International Symposium*, volume 26, pages 512–525. Wiley Online Library, 2016.

[15] Oliver Alt. *Modellbasierte Systementwicklung mit SysML*. Carl Hanser Verlag GmbH Co KG, 2012.

[16] Jeff A Estefan et al. Survey of model-based systems engineering (mbse) methodologies. *Incose MBSE Focus Group*, 25(8):1–12, 2007.

[17] Sanford Friedenthal, Alan Moore, and Rick Steiner. *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[18] Pascal Roques. Mbse with the arcadia method and the capella tool. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, 2016.

[19] Jeff A Estefan and Tim Weilkiens. Mbse methodologies. In *Handbook of model-based systems engineering*, pages 47–85. Springer, 2023.

[20] Mara Fuchs, Yassine Ghanjaoui, Jutta Abulawi, Jörn Biedermann, and Björn Nagel. Enhancement of the virtual design platform for modeling a functional system architecture of complex cabin systems. *CEAS Aeronautical Journal*, 13(4):1101–1117, 2022.

[21] Yassine Ghanjaoui, Mara Fuchs, Jörn Biedermann, and Björn Nagel. Model-based design and multidisciplinary optimization of complex system architectures in the aircraft cabin. *CEAS Aeronautical Journal*, 14(4):895–911, 2023.

[22] Mara Fuchs, Yassine Ghanjaoui, Jörn Biedermann, and Björn Nagel. An approach for linking heterogenous and domain-specific models to investigate cabin system variants. In *INCOSE International Sympo-*

sium, volume 33, pages 1418–1434. Wiley Online Library, 2023.

[23] Jarkko Pakkanen, Tero Juuti, and Timo Lehtonen. Brownfield process: A method for modular product family development aiming for product configuration. *Design Studies*, 45:210–241, 2016.

[24] Constantin Hildebrandt, Aljosha Köcher, Christof Küstner, Carlos-Manuel López-Enríquez, Andreas W Müller, Birte Caesar, Claas Steffen Gundlach, and Alexander Fay. Ontology building for cyber–physical systems: Application in the manufacturing domain. *IEEE Transactions on Automation Science and Engineering*, 17(3):1266–1282, 2020.

[25] Mike Uschold and Robert Jasper. A framework for understanding and classifying ontology applications. In *Proceedings of the IJCAI-99 workshop on ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden*, volume 2, 1999.

[26] Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Science & Business Media, 2006.

[27] Maximilian Weigand and Alexander Fay. Creating virtual knowledge graphs from software-internal data. In *IECON 2022–48th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6. IEEE, 2022.

[28] Pascal Hitzler, Markus Krotzsch, and Sebastian Rudolph. *Foundations of semantic web technologies*. Chapman and Hall/CRC, 2009.

[29] Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, et al. Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21(79):1–31, 2004.

[30] Kristof Meixner, Felix Rinker, Hannes Marcher, Jakob Decker, and Stefan Biffl. A domain-specific language for product-process-resource modeling. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2021.

[31] Niklas Petersen, Lavdim Halilaj, Irlán Grangel-González, Steffen Lohmann, Christoph Lange, and Sören Auer. Realizing an rdf-based information model for a manufacturing company–a case study. In *The Semantic Web–ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II 16*, pages 350–366. Springer, 2017.

[32] Lasse Beers, Maximilian Weigand, Alexander Fay, and Alain Chahine. Entwicklung eines generischen modells für die standardisierte beschreibung von ressourcen in der luftfahrtproduktion. 2022.

[33] Aljosha Köcher, Constantin Hildebrandt, Luis Miguel Vieira da Silva, and Alexander Fay. A formal capability and skill model for use in plug and produce scenarios. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1663–1670. IEEE, 2020.

[34] Aljosha Köcher, Alexander Belyaev, Jesko Hermann, Jürgen Bock, Kristof Meixner, Magnus Volkmann, Michael Winter, Patrick Zimmermann, Stephan Grimm, and Christian Diedrich. A reference model for common understanding of capabilities and skills in manufacturing. *at-Automatisierungstechnik*, 71(2):94–104, 2023.

[35] Jasper H. Bussemaker, Luca Boggero, and Björn Nagel. System architecture design space exploration: Integration with computational environments and efficient optimization. In *AIAA AVIATION 2024 FORUM*, Las Vegas, NV, USA, July 2024.

[36] Jasper H. Bussemaker, Paul Saves, Nathalie Bartoli, Thierry Lefebvre, Rémi Lafage, and Björn Nagel. System architecture optimization strategies: Dealing with expensive hierarchical problems. *Journal of Global Optimization*, 2024. Article submitted.

[37] Jasper H. Bussemaker. ADSG API Reference, 2024. `https://adsg-core.readthedocs.io/en/stable/` [last visited 17.05.2024].

[38] Jasper H. Bussemaker. Sbarchopt: Surrogate-based architecture optimization. *Journal of Open Source Software*, 8(89):5564, 2023.

[39] Object Management Group (OMG). *XML Metadata Interchange (XMI) Specification*, 2014. `http://www.omg.org/spec/XMI/2.4.2` [last visited 06.06.2024].

[40] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.