Enabling Reuse of Knowledge from Coupled Systems in Aerospace Engineering

Ildar Baimuratov[1,2,3,*], Tim Wittenborg[1,2,3], Ihar Antonau[1,4] and Susanna Baars[1,5]

[1]Cluster of Excellence SE2A – Sustainable and Energy-Efficient Aviation, Technical University of Braunschweig, Germany
[2]L3S Research Center, Leibniz University Hannover, Germany
[3]TIB - Leibniz Information Centre for Science and Technology, Hannover, Germany
[4]Institute of structural analysis, Technical University of Braunschweig, Germany
[5]Institute for Acoustics and Dynamics, Technical University of Braunschweig, Germany

Abstract

With the advancement of computational power and physical solvers, the complexity of coupled systems in the aerospace domain will continue to grow. Currently, engineers often model such systems from scratch or reuse local files, despite the potential to formalize, store, and share expert knowledge for broader reuse. In this work, we focus on the open-source software KRATOS Multiphysics, which supports coupled simulations through its CoSimulationApplication. The input JSON files for KRATOS describe deeply nested and interconnected structures, making them difficult to interpret and reuse. To enhance the FAIRness (Findability, Accessibility, Interoperability, and Reusability) of this data, we leverage RDF and OWL standards of the Semantic Web. However, manually constructing semantic graphs and ontologies from such files is both labor-intensive and prone to human error. To address this, we propose an automated procedure for extracting reusable knowledge from coupled simulation data and storing it in a centralized, human- and machine-interpretable knowledge base. Our contribution includes an algorithm that automatically translates KRATOS CoSimulationApplication JSON input files into RDF graphs and derives OWL ontologies from them, capturing generalizable patterns of coupled systems. We validate this approach through a proof-of-concept implementation applied to two representative use cases. Our results demonstrate that the extracted knowledge can be visualized and queried, offering capabilities that were previously achievable only through manual inspection and analysis of JSON files.

Keywords

Coupled modeling, Aerospace engineering, Knowledge extraction, FAIR, OWL

1. Introduction

To advance and explore new aerospace technologies, it is essential to model and solve increasingly complex problems that span multiple disciplines. With ongoing improvements in computational power and advanced physical solvers, the complexity of multidisciplinary models and their analysis is expected to grow significantly. As a result, tools and support systems are needed to help future engineers better understand, visualize, and manage the data generated by their models and simulations [1].

Moreover, the FAIR principles emphasize the need for computational systems to **F**ind, **A**ccess, **I**nteroperate, and **R**euse data with minimal human intervention [2]. In the context of coupled modeling, formalized representations are essential to structure and share domain knowledge effectively. Several data models have been developed for the aerospace engineering, including XML-based formats such as CPACS [3], CMDOWS [4], and KADMOS [5], as well as OWL-based semantic approaches [6, 7]. Despite these advancements, significant challenges remain, particularly limited knowledge sharing and the lack of unified data standards, which hinder the full potential of knowledge-based engineering [8].

1st Workshop on Leveraging SEmaNtics for Transparency in Industrial Systems (SENTIS), co-located with SEMANTiCS'25: International Conference on Semantic Systems, September 3–5, 2025, Vienna, Austria

*Corresponding author.
ildar.baimuratov@l3s.de (I. Baimuratov); tim.wittenborg@l3s.uni-hannover.de (T. Wittenborg); ihar.antonau@tu-braunschweig.de (I. Antonau); s.baars@tu-braunschweig.de (S. Baars)
0000-0002-6573-131X (I. Baimuratov); 0009-0000-9933-8922 (T. Wittenborg); 0000-0002-2432-8663 (I. Antonau); 0009-0003-0327-3143 (S. Baars)

We focus on the open-source software KRATOS Multiphysics [9, 10], which provides a flexible framework for developing multidisciplinary solvers. Coupled simulations within KRATOS are handled by the *CoSimulationApplication* [11], which relies on JSON files as input. These JSON configurations are typically created manually by engineers, embedding expert knowledge that could otherwise be formalized, stored, and reused to define new problems more efficiently. The JSON files often contain deeply nested, interconnected structures that are difficult for new users to interpret, yet include a significant amount of reusable components and recurring patterns. Thus, applying RDF and OWL standards can help align the data with FAIR principles and enhance its machine readability and interoperability.

To enhance accessibility and reuse, we propose a centralized knowledge base that is both human- and machine-interpretable for storing data from coupled simulations. However, manually designing and populating such a knowledge base is time-consuming and prone to human error. To address this, we develop an automated approach for extracting reusable knowledge from previously implemented coupled systems. Specifically, we present an algorithm that converts input JSON files from the KRATOS CoSimulationApplication into RDF format and derives OWL axioms to describe unique classes of coupled systems. We validate this algorithm through a proof of concept, applying it to two distinct use cases, and demonstrate how the proposed approach improves the accessibility and reusability of data in coupled modeling.

The paper is organized as follows: section 2 reviews related work; section 3 provides background on the KRATOS CoSimulationApplication; section 4 describes the proposed method; section 5 presents its application to two use cases; section 6 discusses the results; and finally, section 7 concludes the paper.

2. Related Work

Aerospace engineering makes use of various data models, languages, and ontologies, with XML being the most widely adopted. Vargas-Hernandez et al. [12] employed bond graphs to develop a CAD tool for designing complex electromechanical systems using a standardized taxonomy of elements. Agrawal et al. [13] developed a platform-independent visualization framework for solving multidisciplinary design optimization (MDO) problems. Jackson and Hildreth [14] proposed a flight dynamics model exchange standard. Nagel et al. [15] introduced the Common Parametric Aircraft Configuration Scheme (CPACS), a unified XML-based data model for interfacing between various MDO systems. Munjulury [16] presented RAPID, a relational-based tool for aircraft geometry design. Van Gent [17] introduced the Common MDO Workflow Schema (CMDOWS), a graph-based, application-agnostic formalization of multidisciplinary design analysis and optimization (MDAO) workflows.

Another widely adopted approach is the use of Unified Modeling Language (UML). Lu [18] applied an object-oriented framework to model data for aircraft conceptual design. Oh et al. [19] proposed a mapping methodology to facilitate the exchange of product structure data between CAD and product data management systems. Böhnke [20] compared CPACS [15], STEP[1], and UML in terms of their suitability for data integration in preliminary aircraft design, concluding that a combination of UML or SysML with XML offers the most benefits.

Researchers have also developed specialized software tools to model knowledge in aerospace engineering. Sung & Park [21] introduced a component-based product data management system implemented in Java. Haney [22] developed a data engineering system using Microsoft Access to support aerospace design and forecasting tasks. Herbst [23] presented ADDAM, an object-oriented aircraft design environment implemented in MATLAB. The Python implementation of CMDOWS, known as Knowledge- and Graph-based Agile Design for Multidisciplinary Optimization System (KADMOS), was introduced by Van Gent [17]. Similarly, Risueño et al. [24] developed the MDAO Workflow Design Accelerator (MDAx) in Python, extending the XDSM format with additional design rules.

In this work, we focus on knowledge representation methods with embedded semantics. Using RDF, Bang et al [25] developed Daphne, an intelligent assistant designed to support the architecture of

[1] https://www.iso.org/obp/ui/#iso:std:iso:10303:-1:ed-3:v1:en

Earth-observing satellite systems. Herrero [26] proposed a Digital Thread for airplane design, modeled as a knowledge graph to enable seamless data integration across the design process.

Studies utilizing OWL predominantly rely on manual modeling using the Protégé tool. Curran et al. [27] proposed the Knowledge Optimized Manufacture and Design (KNOMAD) ontology. Kuofie [28] introduced the Rationale-based Design Explanation (RaDEX) framework. Verhagen & Curran [29] developed an ontology for the aerospace composite manufacturing domain. Zhao et al. [30] constructed a knowledge base for spacecraft overall design. More recently, Franzén et al. [7] presented an ontology that captures required functions, design alternatives, and requirements, and applied Description Logic reasoning to generate suitable concepts to fulfill specific functions.

Some studies go a step further by incorporating rule-based reasoning using SWRL. Hoogreef et al. [6] developed an MDO advisory system with decision rules modeled in SWRL. Roelofs & Vos [31] proposed a method for formally describing technologies based on the Basic Formal Ontology, using graph transformation rules to represent logic. Markusheska et al. [32] created an ontology for the automated scheduling of aircraft cabin assembly processes, where SWRL rules are applied to classify and infer data.

The most relevant related work includes methods that incorporate automated knowledge extraction workflows. Dadzie et al. [33] introduced a methodology for extracting knowledge from textual and multimedia documents in the aeronautics domain, using both manual and automated techniques, and representing the information with RDF and OWL DL. Khilwani & Harding [34] developed a framework for managing corporate memory by extracting information from documents, converting it into RDF resources, and identifying relationships using Latent Semantic Analysis. Ezhilarasu & Jennions [35] proposed the Framework for Aerospace Vehicle Reasoning (FAVER), which combines an Adaptive Neuro-Fuzzy Inference System (ANFIS) with causal reasoning. Berquand [36] applied natural language processing (NLP) techniques to extract an ontology of domain-specific models and rules from documents, storing the results in TypeDB[2]. De Leon [37] proposed an approach to enhance association rule mining for Entry, Descent, and Landing datasets by leveraging a user-defined knowledge graph implemented using SysML and Neo4j. However, automated knowledge extraction specifically from coupled systems remains an open and underexplored area.

3. Background

This section provides background on the coupled modeling software that forms the focus of our work. Coupled problems are often addressed using black-box solvers. However, solving multidisciplinary and mixed-fidelity problems through co-simulation requires software that supports efficient coupling strategies, manages the complexity of data exchange between heterogeneous solvers and models, and ensures proper synchronization across multiple processes. KRATOS Multiphysics [9, 10] is a free, open-source framework for the development of multidisciplinary simulation software. It is particularly well-suited to our study, as it has been designed to support complex, coupled simulation scenarios and emphasizes generality, reusability, and extensibility, offering a wide range of tools for implementing finite element applications. It also provides a unified interface for interaction between internal applications and externally controlled solvers, thereby reducing implementation overhead and simplifying data synchronization, especially when compared to traditional distributed data exchange approaches where individual solvers are connected directly to one another.

Co-simulation capabilities in KRATOS are provided by the CoSimulationApplication [11], which manages data exchange and process coordination in coupled simulation environments. This application relies on a JSON file with a specific structure as its input. The core component of the file is the **solver_settings** section, which defines the configuration of the coupled system. Its main elements include:

- **type**: Specifies the coupling method, such as weak or strong coupling using Gauss-Seidel or Jacobi strategies.

[2] https://typedb.com/

- **num_coupling_iterations**: Sets the maximum number of coupling iterations allowed.
- **data_transfer_operators**: Defines how data is transferred between solvers or codes, either through direct copying or using KRATOS mapping.
- **convergence_accelerators**: Used to accelerate convergence in strongly coupled simulations. Available only for strongly coupled solvers.
- **convergence_criteria**: Specifies the criteria for determining when the coupled problem is considered solved. Also only applicable for strongly coupled solvers.
- **coupling_sequence**: Describes the sequence in which disciplines are solved and data is synchronized, establishing the order for initializing and executing the coupling process.
- **input_data_list**: Defines the input data flow. For example, "Discipline 1" receives "data 1" from "Discipline 2", stores it as "data 1", and applies "operation 1".
- **output_data_list**: Specifies the output data flow in a manner similar to the input configuration.
- **solvers**: Lists all solvers or external codes involved in the co-simulation, describing shared meshes, coupled variables or data fields, and communication methods for data exchange.
- **solver_wrapper_settings**: Contains solver-specific parameters, including settings for I/O communication and mesh/data synchronization.

Listing 1 presents a fragment of a typical KRATOS CoSimulation JSON configuration. As shown, the structure is deeply nested and consists of numerous interconnected components. In JSON format, this complexity can be challenging for new users to comprehend. Moreover, the configuration includes a significant amount of reusable structural patterns and recurring entities. Therefore, we propose that a graph-based representation of this data could offer a more intuitive and accessible way to understand and interact with the configuration.

```
{
    "data_transfer_operators": {
        "operation 1": {...}
    },
    "coupling_sequence": [
        {
            "name": "Displine 1",
            "input_data_list": [
                {
                    "data": "data 1",
                    "from_solver": "Displine 2",
                    "from_solver_data": "data 1",
                    "data_transfer_operator": "operation 1"
                }
            ], ...
        },
        {
            "name": "Displine 2", ...
        }
    ],
    "solvers": {
        "Displine 1": {
            "data": {
                "data 1": {...}, ...
            }, ...
        }, ...
    }
}
```

Listing 1: A fragment of the KRATOS CoSimulation configuration in JSON

4. Method

Currently, aerospace engineers typically model coupled systems either from scratch or by reusing locally available files. Our goal is to streamline this process by developing a centralized knowledge base with structured representations, thereby making knowledge from prior coupled simulations more FAIR. This knowledge base includes semantic representations of coupled simulation data in RDF, enhancing interoperability, as well as general knowledge encoded as OWL axioms, which supports human understanding in the formulation and validation of new coupled systems.

Since manual ontology development and knowledge base population are time-consuming and difficult to maintain, our approach is data-driven. While existing tools such as JSON2RDF[3] can convert JSON into RDF, preliminary experiments revealed that these tools fail to accurately capture the underlying data model of the KRATOS CoSimulationApplication and do not support the extraction of generalized knowledge from the data. To address this limitation, we develop an automated workflow that extracts both information and reusable knowledge from KRATOS input JSON files and consistently imports it into the knowledge base. For shortness, we refer to this algorithm as *CoSim2OWL*. This process consists of two main steps: 1) Populating the knowledge base with coupled simulation data extracted from a JSON file and translating it into RDF, and 2) Deriving OWL class axioms that semantically describe the RDF data.

Figure 1 illustrates our proposed approach. On the left, the diagram depicts the current multidisciplinary analysis (MDA) workflow, where aerospace engineers typically model coupled systems from scratch. On the right is the enhanced workflow we propose, which incorporates a knowledge base to support reuse and automation. In our approach, JSON files containing data from previous experiments are processed by a Python module implementing the CoSim2OWL algorithm. This module extracts relevant information and stores it in an OWL-based knowledge base. In subsequent experiments, knowledge at various levels of granularity can be retrieved from the knowledge base via SPARQL queries to support the formulation of new coupled systems. Looking ahead, we plan to further streamline knowledge retrieval by developing an API for the knowledge base and integrating it with existing graphical user interfaces for KRATOS, such as FlowGraph[4]. In the diagram, components highlighted in green represent elements implemented in this work, while dotted elements indicate planned future developments.

4.1. Translating coupled system data into RDF

This subsection describes the first stage of the CoSim2OWL workflow — translating coupled system data into RDF format. The knowledge base, denoted as KB, is initialized with a single predefined class, `coupled_system` (we use teletype font to denote knowledge base objects). This class serves as the superclass for all coupled systems extracted from the CoSimApp JSON files. For each coupled system, the algorithm begins by instantiating the `coupled_system` class with a named individual cs, which corresponds to a specific coupled system described in the input JSON. The algorithm then recursively parses the JSON structure, generating RDF triples and populating KB accordingly. At each iteration, the algorithm constructs an RDF triple using the RDF resource created in the previous step as the subject s, a key from the current JSON object as the predicate name n_p, and the corresponding value as the object v_o. The resulting triple (s, n_p, v_o) is then added to the knowledge base. This recursive traversal continues until the entire JSON structure has been processed and fully represented in RDF.

The object value v_o in a JSON key-value pair can be one of three types: 1) a JSON object, 2) an array, or 3) a literal. Depending on the type, the CoSim2OWL algorithm generates the corresponding RDF triple using different procedures. For clarity and precision, this part of the algorithm is described formally in Algorithm 1. If v_o is a JSON object, it is converted into an `owl:NamedIndividual` o using Algorithm 2. This sub-algorithm also assigns the newly created individual to a specific OWL class C_o, determined based on the predicate name n_p that links v_o to the subject s. The algorithm then

[3] https://github.com/AtomGraph/JSON2RDF

[4] https://github.com/KratosMultiphysics/Flowgraph

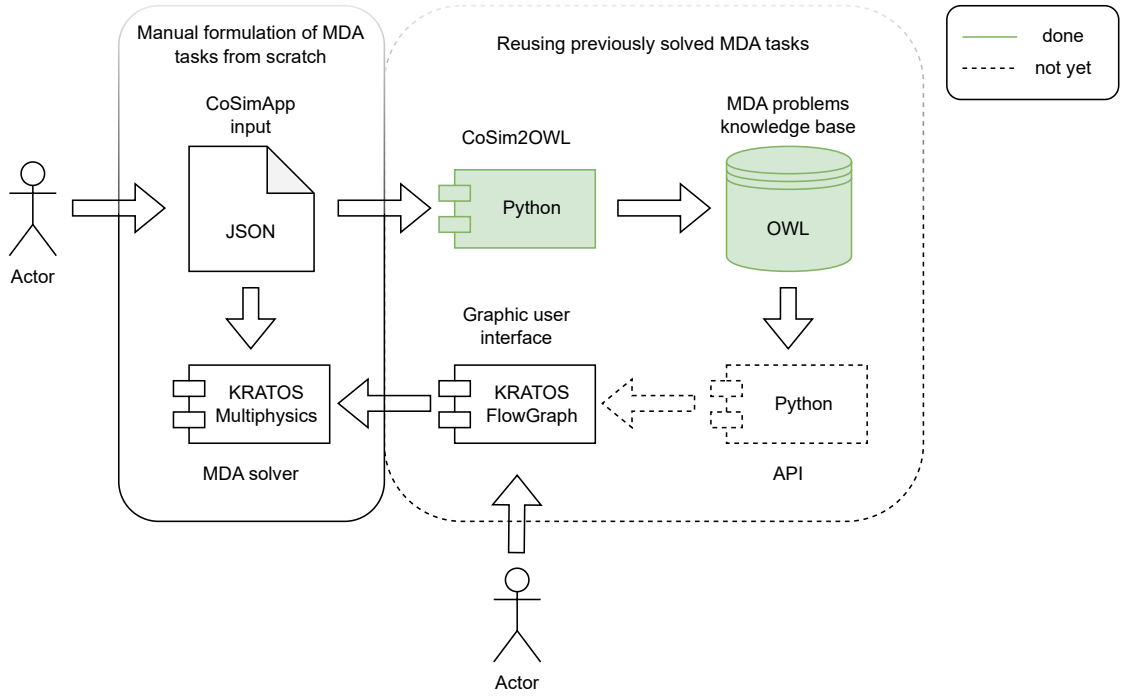


Figure 1: Our approach to enabling the reuse of knowledge from coupled systems

recursively calls Algorithm 1, treating o as the new subject, to process any nested structure and add further RDF triples to the knowledge base. If v_o is a literal, the algorithm either finds or creates an `owl:DataProperty` $prop$ using the predicate name n_p, and adds the triple $(s, prop, v_o)$ to KB. If v_o is an array, the algorithm iterates over each element i in the array. For each item, it checks whether i is a JSON object or a literal and applies the corresponding procedure as described above. Finally, the algorithm returns the current subject s, which can be reused in subsequent iterations or higher-level operations.

Algorithm 1 Triples generation from KRATOS CoSimulation configuration

Require: Knowledge base KB, subject s, predicate name n_p, object value v_o
  if o is a JSON object **then**
    Apply Algorithm 2 to KB, s, n_p, v_o, `owl:Thing`
  else if v_o is an array **then**
    for $i \in v_o$ **do**
      if i is a JSON object **then**
        Get/create an `owl:Class` C_o by n_p
        Apply Algorithm 2 to KB, s, n_p, i, C_o
      else
        Get/create an `owl:DataProperty` $prop$ by n_p
        Add $(s, prop, i)$ to KB
      end if
    end for
  else
    Get/create an `owl:DataProperty` $prop$ by n_p
    Add $(s, prop, v_o)$ to KB
  end if
  return s

Algorithm 2 describes the translation of a JSON object value v_o into an `owl:Named-Individual` o. It is invoked from Algorithm 1 and receives the following inputs: the current knowledge base KB, an existing subject individual s, the object value v_o, the predicate name n_p connecting s to v_o, and a superclass S_o that serves as a parent for any class created for the individual o. Within the algorithm, a class C_o is either retrieved or created as a subclass of S_o, based on the predicate name n_p. The object value v_o is then instantiated as an individual o of class C_o. Using the same predicate name n_p, an `owl:ObjectProperty` rel is retrieved or created to represent the relationship between s and o. The triple (s, rel, o) is then added to the knowledge base KB. Afterward, the algorithm iterates over each key-value pair (k, v) within the JSON object v_o, and recursively calls Algorithm 1 with the updated parameters: the knowledge base KB, the new subject o, key k, and value v. This recursive traversal continues to generate additional RDF triples for the nested structure of v_o.

Algorithm 2 Translating JSON object of KRATOS CoSimulation configuration into named individual

Require: knowledge base KB, subject s, predicate name n_p, object value v_o, object superclass S_o
  Get/create an `owl:Class` C_o by n_p as an
  Get/create an `owl:ObjectProperty` rel by n_p `owl:SubClassOf` S_o
  Instantiate C_o with o
  for $(k, v) \in v_o$ **do**
    Apply Algorithm 1 to KB, o, k, v
    Add (s, rel, o) to KB
  end for

4.2. Describing classes of coupled systems with OWL axioms

Once a specific coupled system cs is represented in the knowledge base KB, we derive general knowledge from it by generating OWL axioms that capture recurring structural patterns as class restrictions. This process is described in Algorithm 3. The algorithm iterates over each individual $a \in KB$. For each individual, we compute the set of predicates P for which a is the subject:

$$pred(a) = \{p : \exists x p(a, x)\}.$$

For each predicate $p \in P$, we collect the set of individuals O such that $p(a, o)$ holds, i.e., the set of objects connected to a via p. We then determine the set of classes C_O to which the individuals in O belong:

$$class(p(a)) = \{C_O : \exists o p(a, o), o \in C_O\}.$$

For each class $C_o \in C_O$, we count the number k of individuals in O that belong to C_o. Then, for each class C_i that a is an instance of, we generate OWL class axioms as follows: if $k = 1$, we add an existential restriction:

$$C_i \sqsubseteq \exists p.C_o,$$

otherwise, we add a cardinality restriction:

$$C_i \sqsubseteq = k p.C_o.$$

To summarize, for each key-value pair (k, v) in the JSON object extracted from the input file of the KRATOS CoSimulationApplication, we create an individual cs in the knowledge base KB to represent the coupled system under consideration. The key k and value v are then passed to Algorithm 1, which recursively traverses the structure of v and adds corresponding RDF triples to KB. Finally, we apply Algorithm 3 to KB in order to enrich the knowledge base with OWL axioms that capture generalizable patterns and semantic constraints. These axioms generalize the observed patterns in the RDF data into reusable, machine-interpretable OWL semantics, thereby enriching the knowledge base with domain-level conceptual knowledge. The described CoSim2OWL algorithm has been implemented in Python using the Owlready2 library [38]. The implementation is publicly available on GitHub[5].

[5] https://anonymous.4open.science/r/coupled_modelling-3909/

Algorithm 3 Generation of OWL axioms describing classes of coupled systems

Require: KB
  for $a \in KB$ **do**
    $C = class(a)$
    $P = pred(a)$
    for $p \in P$ **do**
      $O = class(p(a))$
      for $C_o \in C_O$ **do**
        $k = count(C_o, class(p(a))$
        for $C_i \in C$ **do**
          if $k == 1$ **then**
            $C_i \sqsubseteq \exists p.C_o$
          else
            $C_i \sqsubseteq= kp.C_o$
          end if
        end for
      end for
    end for
  end for

5. Use Cases

To validate the proposed workflow, we examine two case studies of coupled systems from the aerospace engineering domain. Both cases involve fluid-structure interaction (FSI) problems but differ in the type of models used: 1) analytical functions and 2) finite element-based physical models (FEM). The developed prototype is applied to extract a knowledge base from each of these coupled systems.

5.1. Analytical Functions

The first use case involves a simplified aero-structural system consisting of a two-dimensional airfoil mounted on a spring. This airfoil–spring system is modeled using an analytical approach originally developed by Sobieszczanski-Sobieski in [39], see Figure 2.

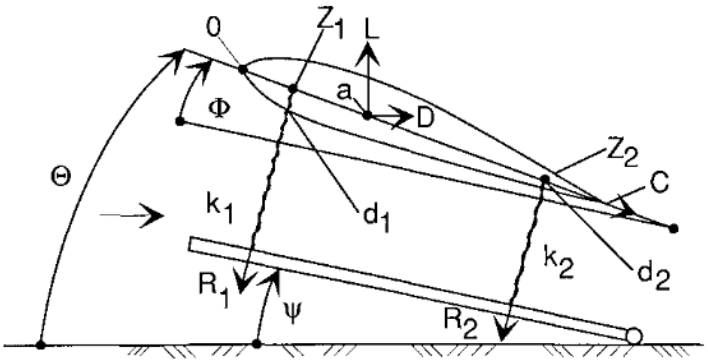


Figure 2: An analytical aerodynamics-structures coupled system [39]

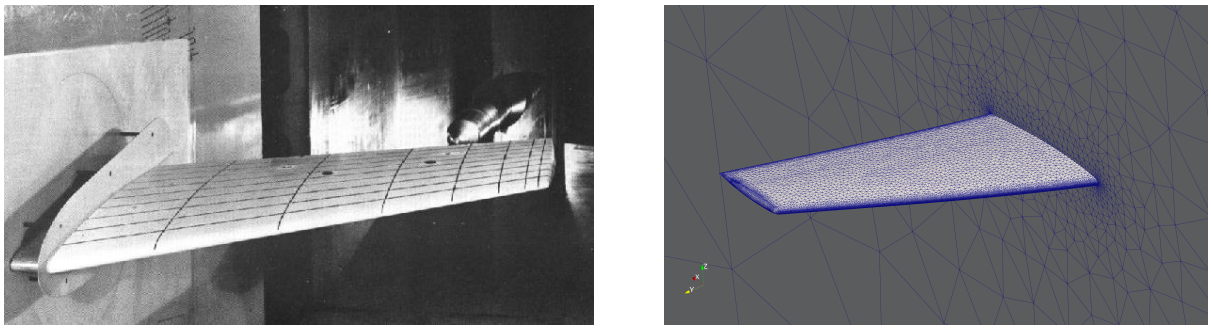
The analytical model consists of two coupled disciplines: 1) the fluid dynamics around the airfoil and 2) the structural behavior of the supporting spring system. These disciplines are coupled through two shared variables: the elastic support pitch angle of the airfoil (Φ) and the resulting lift force (L). The model operates according to the following process:

1. The lift force acting on the airfoil is computed using aerodynamic equations and is directly transferred to the structural model by simple data copying.
2. The deformation of the spring system caused by the applied lift force L alters the elastic support pitch angle Φ, which is then passed back to the aerodynamic model. This updated pitch angle is used to recompute the lift force.
3. The process repeats iteratively until a stationary point is reached.

Due to the assumed nonlinear relationship between the lift coefficient and the pitch angle, the solution cannot be obtained analytically in a single step. Instead, it requires iterative resolution using a strong coupling strategy.

5.2. FE-Based Physical Models

The second use case is based on a finite element model of an aeroelastic ONERA M6 wing, see Figure 3. The ONERA M6 wing is a well-established benchmark case for validating computational fluid dynamics (CFD) methods in external aerodynamic flows. Its relatively simple geometry, combined with the presence of complex transonic phenomena, such as local supersonic regions, shock waves, and turbulent boundary layer separation, makes it particularly suitable for such evaluations. Due to its extensive use in the literature as a reference case [40], the ONERA M6 wing has become a de facto standard for assessing the accuracy and performance of CFD codes.



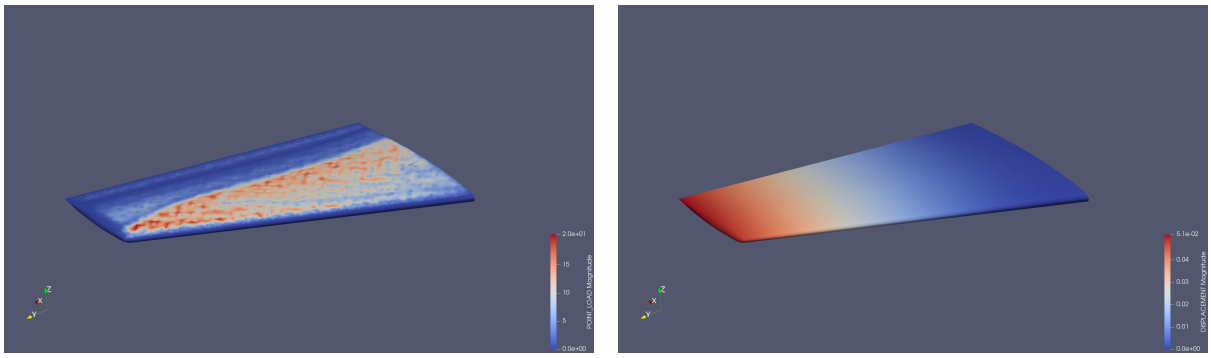
(a) photo from [40]

(b) numerical CFD mesh

Figure 3: An aeroelastic ONERA M6 wing

The open-source SU2 solver [41] is used to solve the steady 3D Euler equations and compute the aerodynamic forces of the CFD model. The structural mechanics (SM) model shares the same geometry as the ONERA M6 wing and is discretized using solid tetrahedral finite elements. It is solved using the KRATOS StructuralMechanicsApplication. The FE-based coupled model can be described as follows:

1. The coupled problem is solved using the KRATOS CoSimulationApplication, which employs a strong Gauss–Seidel coupling strategy as specified in the **type** property of the input JSON file. To accelerate convergence and reduce computational cost, convergence acceleration methods are defined in the **convergence_accelerators** section.
2. The CFD and SM models are remotely orchestrated by the CoSimulationApplication and executed sequentially according to the order specified in **coupling_sequence**.
3. The SU2 solver, defined in the **solvers** section, computes the aerodynamic solution and outputs the lift force.
4. The lift force is transferred from the CFD model to the SM model using a mapping strategy provided by the KRATOS MappingApplication, as specified in the **data_transfer_operators** property, see Figure 4a.
5. The KRATOS StructuralMechanicsApplication, also defined in **solvers**, computes the resulting structural displacements in response to the transferred aerodynamic loads, see Figure 4b.
6. This process is repeated iteratively until convergence is reached, as defined by the criteria specified in **convergence_criteria**.



(a) lift force (b) structural displacements

Figure 4: Coupled variables on ONERA M6 wing

5.3. Results

To extract a knowledge base from the described coupled systems, we process the input JSON files corresponding to each case and create an instance of the `coupled_system` class for each. The instance for the first use case is labeled `low_fid_models`, while the second use case is represented by the instance `Onera_FSI`. We then run our prototype to translate the JSON data into a graph-based representation and to derive semantic knowledge from it. The resulting knowledge base contains 549 axioms, 41 classes, 41 individuals, 32 data properties, and 13 object properties. Figure 5 visualizes the part of the knowledge base corresponding to the `low_fid_models` coupled system.

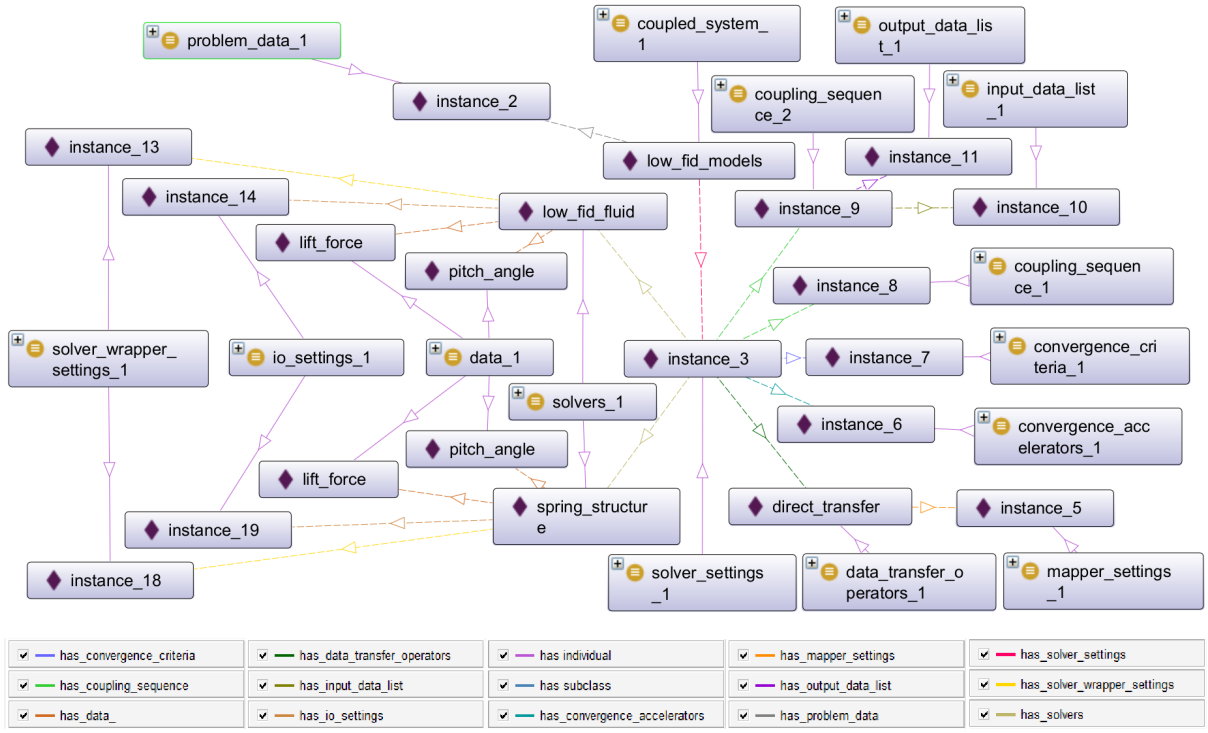
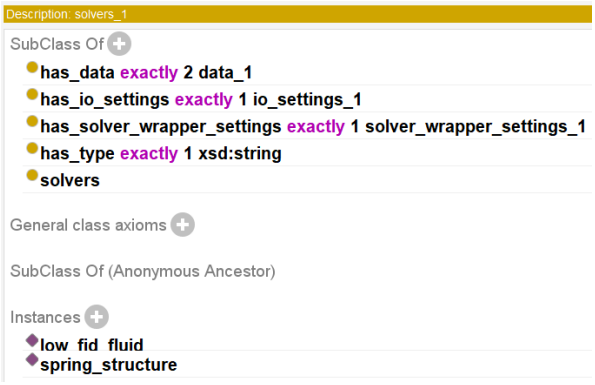


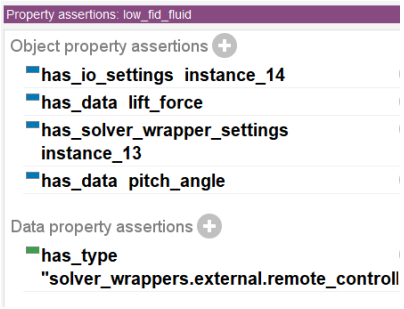
Figure 5: Visualization of the `low_fid_models` coupled system in the knowledge base

As an example of the generated axioms, Figure 6a illustrates the properties of the `solvers_1` class, which represents both the `low_fid_fluid` and `spring_structure` solver instances. This class is a subclass of `solvers` and is required to be connected to two instances of the `data_1` class, one instance of `io_settings_1`, and one instance of `solver_wrapper_settings_1`. Additionally, it possesses

a string value for the `has_type` property. Figure 6b shows the properties of the `low_fid_fluid` individual.



(a) Properties of a generated class



(b) Properties of a generated instance

Figure 6: Generated knowledge base entities

As an example of how the knowledge base can be utilized, Listing 2 demonstrates a SPARQL query that retrieves all solvers used in previous experiments. This query returns solvers such as `low_fid_fluid` and `spring_structure` from the first coupled system, as well as `SM` and `CFD` from the second coupled system. This information previously could only be obtained by manually collecting and inspecting local JSON files.

```
SELECT DISTINCT ?solver
WHERE {
        ?solver rdf:type ?solvers .
        ?solvers rdfs:subClassOf kb:solvers .
}
-----------------------------------------------------------
spring_structure
low_fid_fluid
SM
CFD
```

Listing 2: SPARQL query for retrieving solver instances across all coupled systems in the knowledge base

6. Discussion

6.1. Improvements over existing conversion tools

In this subsection, we demonstrate that existing tools for converting JSON data into RDF are not sufficient for our goals. Listing 3 shows RDF in TTL format generated by the JSON2RDF tool based on the airfoil–spring system, while Listing 4 presents RDF generated by our approach using the same content. The two outputs differ in both structure and semantic expressiveness. The RDF generated by JSON2RDF contains blank nodes, which require deep path traversal when querying. In contrast, our tool creates named individuals, giving each resource an identifiable IRI. The output from JSON2RDF is also harder to maintain and reuse, as it is less modular. Finally, the RDF produced by our approach explicitly employs OWL classes and properties, enabling more advanced semantic modeling.

```
...
:solver_settings [
    :convergence_accelerators [
        :data_name "pitch_angle";
        :solver "low_fid_fluid" ;
        :type "aitken"
    ] ;
    :data_transfer_operators [ :
        direct_transfer [
        :mapper_settings [
          :mapper_type "
              nearest_neighbor" ;
          :echo_level "3"^^xsd:int
        ] ;
        :type "kratos_mapping"
      ] ] ;
    :echo_level "3"^^xsd:int ;
    :num_coupling_iterations "20"^^xsd:
        int ;
    :type "coupled_solvers.
        gauss_seidel_strong" ;
...
```

Listing 3: RDF generated by JSON2RDF

```
...
:instance_3 rdf:type owl:NamedIndividual ,
        :solver_settings_1 ;
    :has_convergence_accelerators :instance_6 ;
    ...
    :has_data_transfer_operators :instance_4 ;
    ...
    :has_echo_level 3 ;
    :has_num_coupling_iterations 20 ;
    :has_type "coupled_solvers.
        gauss_seidel_strong" .
...
:instance_6 rdf:type owl:NamedIndividual ,
        :convergence_accelerators_1 ;
    :has_data_name "pitch_angle" ;
    :has_solver "low_fid_fluid" ;
    :has_type "aitken" .
...
:instance_4 rdf:type owl:NamedIndividual ,
        :data_transfer_operators_1 ;
    :has_mapper_settings :instance_5 ;
    :has_type "kratos_mapping" ;
    rdfs:label "direct_transfer" .
...
```

Listing 4: RDF generated by our approach

6.2. Advancing FAIRness of coupled simulation data

In the current state of our approach, we do not perform a user evaluation, as we do not expect aerospace engineers to interact directly with Semantic Web tools such as Protégé. We plan to conduct a user study after integrating our approach with user-friendly interfaces like FlowGraph. In the meantime, we provide an analysis of the FAIRness of the generated representation in comparison to the source JSON data (see Table 1). This comparison shows that transforming coupled simulation configurations from JSON into an ontology-based RDF representation makes a significant advancement in terms of FAIR compliance. Whereas the original JSON files are functional only within their specific software ecosystem and serve as siloed containers with implicit semantics, our approach transforms the configuration data into a FAIR digital object. Each piece of information is assigned a unique identifier, embedded within a formal, machine-interpretable schema, and linked through explicit, typed relationships. Moreover, the proposed approach enhances data reusability and interoperability by reusing previously generated classes and properties when modeling new coupled systems. For example, see Listing 5 and Listing 6, which describe two instances, one from the `low_fid_fluid` model and the other from the `Onera_FSI` model. Both instances belong to the same class, `problem_data_1`, as they share the exact same set of properties and differ only in their echo level value. This ensures structural consistency and results in a unified MDA knowledge base.

```
:instance_2 rdf:type owl:NamedIndividual ,
        :problem_data_1 ;
    :has_parallel_type "OpenMP" ;
    :has_echo_level 0 ;
    :has_end_time 1.0 ;
    :has_print_colors "true"^^xsd:boolean ;
    :has_start_time 0.0 .
```

Listing (5) An instance of problem data from the `low_fid_fluid` model

```
:instance_44 rdf:type owl:NamedIndividual ,
        :problem_data_1 ;
    :has_parallel_type "OpenMP" ;
    :has_echo_level 2 ;
    :has_end_time 1.0 ;
    :has_print_colors "true"^^xsd:boolean ;
    :has_start_time 0.0 .
```

Listing (6) An instances of problem data from the `Onera_FSI` model

Figure 7: Individuals from different coupled systems assigned to the same class

Table 1
Analysis of FAIR Compliance of the Generated Results

Source JSON	RDF/OWL representation
Findable (F)	
F1: (Meta)data are assigned a globally unique and persistent identifier	
JSON keys are context-dependent and not globally unique	Each RDF resource has a unique URI
F2: Data are described with rich metadata	
Keys lack formal semantics	Ontology defines each entity as an instance of a class with restrictions, specifying exactly which properties it must have, types of their values, and their cardinality
F3: Metadata clearly and explicitly include the identifier of the data they describe	
No explicit metadata-to-data linking	Metadata formally link data to URIs; inherent in RDF model
F4: (Meta)data are registered or indexed in a searchable resource	
JSON is not inherently searchable without ingesting into a specific search engine (e.g., Elasticsearch) that is configured to understand its structure	RDF can be loaded into triple stores and queried via standard language (SPARQL)
Accessible (A)	
A1: (Meta)data are retrievable via standardized protocols	
Retrieving a specific piece of data within the file requires custom parsing that understands the JSON structure. For example, you cannot retrieve `num_coupling_iterations` by its name alone	Any piece of data can be queried via HTTP and SPARQL endpoints
A2: Metadata are accessible even if data are unavailable	
JSON stores metadata (keys) and data (values) in the same file. All metadata is lost if file is deleted	Ontology remains accessible independently of instance data. One could know what a `coupled_system_1` class is (its constraints) even if no instances of it currently exist
Interoperable (I)	
I1: Uses formal, shared, machine-readable language	
JSON lacks formal semantics	RDF/OWL are W3C standards with formal semantics
I2: Uses FAIR-compliant vocabularies	
Ad-hoc keys without standard vocabularies	Uses standard vocabularies like `rdf:`, `owl:`, etc. Its own vocabulary is explicitly defined within the ontology, making it a new, FAIR vocabulary that others could reuse
I3: Includes qualified references to other (meta)data	
References are string-based and internal	Links are URI-based, typed, and globally resolvable
Reusable (R)	
R1: Rich description with accurate and relevant attributes	
Keys and values lack context, constraints, and reuse mechanisms	OWL class definitions with `owl:Restrictions` define cardinalities and value types

7. Conclusion

In this research, we addressed the challenge of enabling the reuse of expert knowledge for modeling coupled systems by representing it using RDF and OWL languages. We developed an automated workflow to extract knowledge from KRATOS CoSimulationApplication input JSON files and store it in a centralized knowledge base. As a result, our approach enhances the accessibility and interoperability of MDA data and enables the extraction of unique, human- and machine-interpretable general patterns of coupled systems. We validated our method through two use cases and demonstrated how the extracted

knowledge can be effectively visualized and queried, while previously accessing such knowledge required manual collection and analysis of JSON files.

Although our work represents only an initial step toward data FAIRification in the aerospace engineering domain, addressing this challenge is essential to manage the imminent overflow of technical data that can no longer be handled manually. In future work, we will validate the scalability of our approach and further enhance the FAIRness of the data and knowledge stored in our knowledge base. Specifically, we plan to further increase its FAIRness by synchronizing our resources with established knowledge bases such as Wikidata and DBpedia. Additionally, we aim to extend the knowledge base with a Python API to facilitate knowledge retrieval and integrate it with existing graphical user interfaces.

Acknowledgments

We would like to acknowledge the funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany´s Excellence Strategy – EXC 2163/1 - Sustainable and Energy Efficient Aviation – Project-ID 390881007 and the German Ministry of Education and Research (BmBF) for the project KISSKI AI Service Center (01IS22093C).

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT in order to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] I. Antonau, S. Warnakulasuriya, S. Baars, I. Baimuratov, T. Wittenborg, L. Kreuzeberg, A. Attravanam, R. Wüchner, Challenges in realizing 3rd generation multidisciplinary design optimization, Advances in Computational Science and Engineering 5 (2025) 1–21.

[2] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. 't Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, B. Mons, The FAIR guiding principles for scientific data management and stewardship, Scientific Data 3 (????) 160018. URL: https://doi.org/10.1038/sdata.2016.18. doi:10.1038/sdata.2016.18.

[3] M. Alder, E. Moerland, J. Jepsen, B. Nagel, Recent advances in establishing a common language for aircraft design with cpacs, 2020.

[4] I. van Gent, G. La Rocca, M. Hoogreef, Cmdows: A proposed new standard to store and exchange mdo systems, 2017.

[5] I. van Gent, G. La Rocca, L. Veldhuis, Composing mdao symphonies: graph-based generation and manipulation of large multidisciplinary systems, 2017. doi:10.2514/6.2017-3663.

[6] M. F. Hoogreef, R. d'Ippolito, R. Augustinus, G. La Rocca, A multidisciplinary design optimization advisory system for aircraft design, in: 5th CEAS Air and Space Conference" Challenges in European Aerospace, 2015. URL: https://repository.tudelft.nl/islandora/object/uuid:8beb86d6-40a7-4c03-bc18-8512bb5bb548/datastream/OBJ/download;A.

[7] L. Knöös Franzén, R. C. Munjulury, P. Krus, Ontology-assisted aircraft concept generation, in: 33rd Congress of the International Council of the Aeronautical Sciences, ICAS 2022, Stockholm, Sweden, 4-9 September, 2022, volume 2, International Council of the Aeronautical Sciences (ICAS), 2022, pp. 1510–1526.

[8] T. Wittenborg, I. Baimuratov, L. K. Franzén, I. Staack, U. Römer, S. Auer, Knowledge-based aerospace engineering–a systematic literature review, arXiv preprint arXiv:2505.10142 (2025). URL: https://doi.org/10.48550/arXiv.2505.10142. doi:10.48550/arXiv.2505.10142.

[9] P. Dadvand, R. Rossi, E. Oñate, An Object-oriented Environment for Developing Finite Element Codes for Multi-disciplinary Applications, Archives of Computational Methods in Engineering 17 (2010) 253–297. URL: http://link.springer.com/10.1007/s11831-010-9045-2. doi:10.1007/s11831-010-9045-2.

[10] V. M. Ferrándiz, P. Bucher, R. Zorrilla, R. Rossi, A. Cornejo, Jcotela, M. A. Celigueta, J. Maria, Tteschemacher, C. Roig, M. Masó, Suneth Warnakulasuriya, G. Casas, M. Núñez, P. Dadvand, S. Latorre, I. De Pouplana, J. I. González, F. Arrufat, Riccardotosi, AFranci, A. Ghantasala, P. Wilson, Dbaumgaertner, Bodhinanda Chandra, A. Geiser, K. B. Sautter, Inigo Lopez, Lluís, Javi Gárate, KratosMultiphysics/Kratos: Release 9.3, 2023. URL: https://zenodo.org/record/3234644. doi:10.5281/ZENODO.3234644, language: en.

[11] P. Bucher, A. Ghantasala, P. Dadvand, R. Wüchner, K.-U. Bletzinger, Realizing cosimulation in and with a multiphysics framework, 2021. doi:10.23967/coupled.2021.048.

[12] N. Vargas-Hernandez, J. Shah, Z. Lacroix, Development of a computer aided conceptual design tool for complex electromechanical systems, in: Computational Synthesis: From Basic Building Blocks to High Level Functionality, Papers from the 2003 AAAI Symposium Technical Report SS-03-02, 2003, pp. 255–261.

[13] G. Agrawal, S. Parashar, K. W. English, C. L. Bloebaum, Web-based visualization framework for decision-making in multidisciplinary design optimization, in: 45th AIAA/ASME Conference, Citeseer, 2004, pp. 19–22. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e94eda1e9ac59fb4202d213685594c903e7910d7.

[14] E. B. Jackson, B. L. Hildreth, Progress Toward a Format Standard for Flight Dynamics Models, 2006. URL: https://ntrs.nasa.gov/citations/20060048508.

[15] B. Nagel, D. Böhnke, V. Gollnick, P. Schmollgruber, A. Rizzi, G. La Rocca, J. J. Alonso, Communication in aircraft design: Can we establish a common language, in: 28th International Congress of the Aeronautical Sciences, volume 201, 2012. URL: https://www.icas.org/ICAS_ARCHIVE/ICAS2012/PAPERS/201.PDF, issue: 2.

[16] V. R. C. Munjulury, Knowledge based integrated multidisciplinary aircraft conceptual design, PhD Thesis, Linköping University Electronic Press, 2014. URL: https://www.diva-portal.org/smash/record.jsf?pid=diva2:719775.

[17] I. van Gent, Agile MDAO systems: a graph-based methodology to enhance collaborative multidisciplinary design, 2019. URL: https://research.tudelft.nl/en/publications/agile-mdao-systems-a-graph-based-methodology-to-enhance-collabora.

[18] Z. Lu, Data management in an object-oriented distributed aircraft conceptual design environment, PhD Thesis, Citeseer, 2005. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=dca1203c612b3e35449e180cb605431b8b627511.

[19] Y. Oh, S.-h. Han, H. Suh, Mapping product structures between cad and pdm systems using uml, Computer-aided design 33 (2001) 521–529.

[20] D. Böhnke, Data integration in preliminary airplane design, PhD Thesis, 2009. URL: https://elib.dlr.de/59735/.

[21] C. S. Sung, S. J. Park, A component-based product data management system, The International Journal of Advanced Manufacturing Technology 33 (2007) 614–626.

[22] E. Haney, Data Engineering in Aerospace Systems Design & Forecasting, PhD Thesis, 2016. URL: https://rc.library.uta.edu/uta-ir/handle/10106/25877.

[23] S. Herbst, Development of an aircraft design environment using an object-oriented data model in MATLAB, PhD Thesis, Technische Universität München, 2018. URL: https://mediatum.ub.tum.de/1431402.

[24] R. A.P, B. J.H, C. P.D, N. B, MDAx: Agile Generation of Collaborative MDAO Workflows for Complex Systems, 2020. URL: https://zenodo.org/record/4672051. doi:10.2514/6.2020-3133.

[25] H. Bang, A. Virós Martin, A. Prat, D. Selva, Daphne: An Intelligent Assistant for Architecting Earth Observing Satellite Systems, in: 2018 AIAA Information Systems-AIAA Infotech @ Aerospace, American Institute of Aeronautics and Astronautics, Kissimmee, Florida, 2018. URL: https://arc.aiaa.org/doi/10.2514/6.2018-1366. doi:10.2514/6.2018-1366.

[26] J. Herrero, A System Architecture for the Digital Thread in Commercial Airplane Design, PhD Thesis, Massachusetts Institute of Technology, 2022. URL: https://dspace.mit.edu/handle/1721.1/143229.

[27] R. Curran, W. Verhagen, M. Van Tooren, The KNOMAD Methodology for Integration of Multidisciplinary Engineering Knowledge Within Aerospace Production, in: 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics, Orlando, Florida, 2010, p. 1315. URL: https://arc.aiaa.org/doi/10.2514/6.2010-1315. doi:10.2514/6.2010-1315.

[28] E. J. Kuofie, Radex: A rationale-based ontology for aerospace design explanation, 2010. URL: http://essay.utwente.nl/59926/.

[29] W. J. Verhagen, R. Curran, Ontological modelling of the aerospace composite manufacturing domain, in: Improving Complex Systems Today: Proceedings of the 18th ISPE International Conference on Concurrent Engineering, Springer, 2011, pp. 215–222.

[30] H. Zhao, Z. Sun, H. Zhang, Application of knowledge engineering in spacecraft overall design, in: Journal of Physics: Conference Series, volume 1510, IOP Publishing, 2020, p. 012015. URL: https://iopscience.iop.org/article/10.1088/1742-6596/1510/1/012015/meta, issue: 1.

[31] M. N. Roelofs, R. Vos, Automatically inferring technology compatibility with an ontology and graph rewriting rules, Journal of Engineering Design 32 (2021) 90–114. URL: https://www.tandfonline.com/doi/full/10.1080/09544828.2020.1860202. doi:10.1080/09544828.2020.1860202, publisher: Taylor & Francis.

[32] N. Markusheska, V. Srinivasan, J.-N. Walther, A. Gindorf, J. Biedermann, F. Meller, B. Nagel, Implementing a system architecture model for automated aircraft cabin assembly processes, CEAS Aeronautical Journal 13 (2022) 689–703. URL: https://link.springer.com/10.1007/s13272-022-00582-6. doi:10.1007/s13272-022-00582-6, publisher: Springer.

[33] A.-S. Dadzie, R. Bhagdev, A. Chakravarthy, S. Chapman, J. Iria, V. Lanfranchi, J. Magalhães, D. Petrelli, F. Ciravegna, Applying semantic web technologies to knowledge sharing in aerospace engineering, Journal of Intelligent Manufacturing 20 (2009) 611–623. URL: http://link.springer.com/10.1007/s10845-008-0141-1. doi:10.1007/s10845-008-0141-1.

[34] N. Khilwani, J. A. Harding, Managing corporate memory on the semantic web, Journal of Intelligent Manufacturing 27 (2016) 101–118. URL: http://link.springer.com/10.1007/s10845-013-0865-4. doi:10.1007/s10845-013-0865-4, publisher: Springer.

[35] C. M. Ezhilarasu, I. K. Jennions, Development and implementation of a framework for aerospace vehicle reasoning (faver), IEEE Access 9 (2021) 108028–108048.

[36] A. Berquand, Text mining and natural language processing for the early stages of space mission design, 2021. URL: https://stax.strath.ac.uk/concern/theses/xw42n836q.

[37] S. S. De Leon, Intelligent Data Understanding for Entry, Descent, and Landing, Architecture Analysis, PhD Thesis, Texas A&M University, 2021. URL: https://search.proquest.com/openview/c19723dd75db58eaa895a2026b5d308b/1?pq-origsite=gscholar&cbl=18750&diss=y.

[38] J.-B. Lamy, Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies, Artificial Intelligence in Medicine 80 (2017) 11–28. URL: https://www.sciencedirect.com/science/article/pii/S0933365717300271. doi:https://doi.org/10.1016/j.artmed.2017.07.002.

[39] J. Sobieszczanski-Sobieski, Sensitivity of complex, internally coupled systems, AIAA Journal 28 (1990) 153–160. URL: https://arc.aiaa.org/doi/10.2514/3.10366. doi:10.2514/3.10366.

[40] ONERA M6 Wing, 2021. URL: https://www.grc.nasa.gov/www/wind/valid/m6wing/m6wing.html.

[41] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, J. J. Alonso, Su2: An open-source suite for multiphysics simulation and design, Aiaa Journal 54 (2016) 828–846.