33RD CONGRESS
OF THE INTERNATIONAL COUNCIL
OF THE AERONAUTICAL SCIENCES
STOCKHOLM, SWEDEN, 4–9 SEPTEMBER, 2022

ICAS 2022 SWEDEN

RULE-BASED VERIFICATION OF A GEOMETRIC DESIGN USING THE CODEX FRAMEWORK

Brigitte Boden[1], Yannic Cabac[1], Tim Burschyk[1], Björn Nagel[1]

[1]Institute of System Architectures in Aeronautics, DLR, Hein-Saß-Weg 22, 21129 Hamburg

Abstract

The definition of requirements is an important step of the aircraft design process. Usually, experts of different domains are involved in this multi-disciplinary process. Hence, knowledge from the different domains has to be efficiently shared and integrated. Supporting this process is the aim of the Codex (COllaborative DEsign and eXploration) framework, which is a Knowledge-Based Engineering (KBE) system currently being developed at the German Aerospace center (DLR).

The design process results in the aircraft geometry determination which can be challenging for some components, subsystems or even new designs. To support this engineering task, this paper presents the *codex-geometry* module for geometry modelling using semantic web technologies within the Codex KBE framework.

User-defined rules are used to express geometric requirements and analysis functions, which are automatically executed in order to verify and evaluate a geometric design. The presented approach is demonstrated by the example use case of a conventional fuel system model including pipe systems and fuel pumps.

Keywords: Fuel System, Geometry, Requirements, Semantic Web, Knowledge-Based Engineering

1. Introduction

The description of aircraft geometry is one of the main challenges during the overall aircraft design process. The design process starts with the requirements definition and further involves three major phases: Conceptual design phase, preliminary design phase and detailed design phase [1]. During the conceptual and preliminary design phase the geometry is usually described by a set of common aircraft parameters. Among others, such geometric parameters are subsequently used to derive aircraft characteristics mainly based on empiric methods. This approach works well for conventional aircraft concepts since comprehensive aircraft data and thus methods are available. However, for unconventional aircraft concepts this design approach can be challenging due to a lack of data. Especially the integration of unconventional onboard systems or propulsion systems into the aircraft design requires new methods for handling more detailed geometries. Therefore, this paper presents an approach to address the geometric integration of new subsystems of any kind at an early design stage.

Another important aspect of the aircraft design process is that it includes a large number of experts from different domains or disciplines, who require effective ways of sharing and integrating their knowledge. Modern Knowledge-Based Engineering (KBE) systems support this by providing highly specialized environments and languages which are fit to their respective domain of application. However, this poses limitations on the integration of knowledge between the different domains, which

can lead to errors and inconsistencies among the different models. The use of Semantic Web Technologies (SWT) delivers a domain-neutral way of knowledge formalization and data integration. This can drastically reduce the effort required to integrate knowledge of multiple domains into a single representation. The COllaborative DEsign and eXploration (Codex) framework [2], which is currently being developed at the German Aerospace center (DLR), aims at combining KBE and SWT technologies for the continued digitalization of the design process. The framework can be used to create domain-specific knowledge-bases and integrate them into a single model of the overall product.

This paper presents the *codex-geometry* module, which is developed as part of the Codex framework. The scope of this work includes geometry modelling using semantic web technologies as well as the formulation of geometric requirements and analysis functions. Geometric requirements are used in order to verify the implemented geometry model, and analysis functions are applied to prepare the design assessment for future developments. A simplified conventional fuel system model, including various pipe systems and fuel pumps, acts as an example use case to evaluate the presented approach. It is chosen due to the following three reasons:

- There is comprehensive design experience for conventional fuel systems in literature which can be used for its application and testing
- The geometric modelling of the fuel system is essential to meet system requirements at an early design stage [3]
- The fuel system including its complex piping is appropriate to evaluate the capabilities of codex-geometry

1.1 Related and Present Work

Two commonly used KBE tools in the aerospace sector are Pacelab APD [4] and ParaPy [5]. In both tools, the data is modeled in a hierarchical model created in an object-oriented programming language. This is effecive when modeling a product and sub-products from the same domain, but can get complicated to use when integrating the knowledge from different discplines (and therefore, different models). In contrast, the Codex framework follows a non-hierarchical modeling approach which focuses on avoiding such problems. It is based on "semantically expressing the precise meaning between models of different domains, which we assume to be key to create a highly collaborative KBE application" [2].

In [6] and [7] the creation of graph-based design languages or "design grammars" are discussed for a satellite design process. Design Cockpit 43 [8] aims at automating the design process and uses graph-based design languages to store the engineering knowledge. [9] presents an algorithm for the automated generation of pipe routes in a given installation space based on a design language. Those languages are based on UML (Unified modeling language) and thus on an object-oriented model, whereas in the approach presented here a non-hierarchical, semantic web-based approach is followed.

Some of the features of *codex-geometry* were already used in [10] for modelling and volume computations of a liquid hydrogen tank.

The rest of this paper is structured as follows: Section 2 generally discusses the kind of fuel system model that is used as a demonstrative model. Section 3 briefly describes the Codex framework and presents the new *codex-geometry* module. Section 4 discusses the present approach to model the fuel system as well as the requirements. In Section 5 the results of the present approach are shown by a parametric study. Section 6 gives a short outlook on future work.

2. Fuel System Model

Today's commercial large aircraft (certified according to CS25) are equipped with complex pump driven fuel systems [11,12]. Such systems have the following basic functions:

- Engine and auxiliary power unit (APU) fuel feeding
- Fuel storage at suitable pressure
- Fuel transfer between tanks
- Refueling and defueling
- Jettison (not applied to all aircraft)
- Fuel measurement and management

Each required fuel system function is fulfilled by one or more subsystem(s). A detailed breakdown is given by the ATA chapter 28-00 [13]. Figure 1 gives a schematic overview of a typical fuel system architecture.

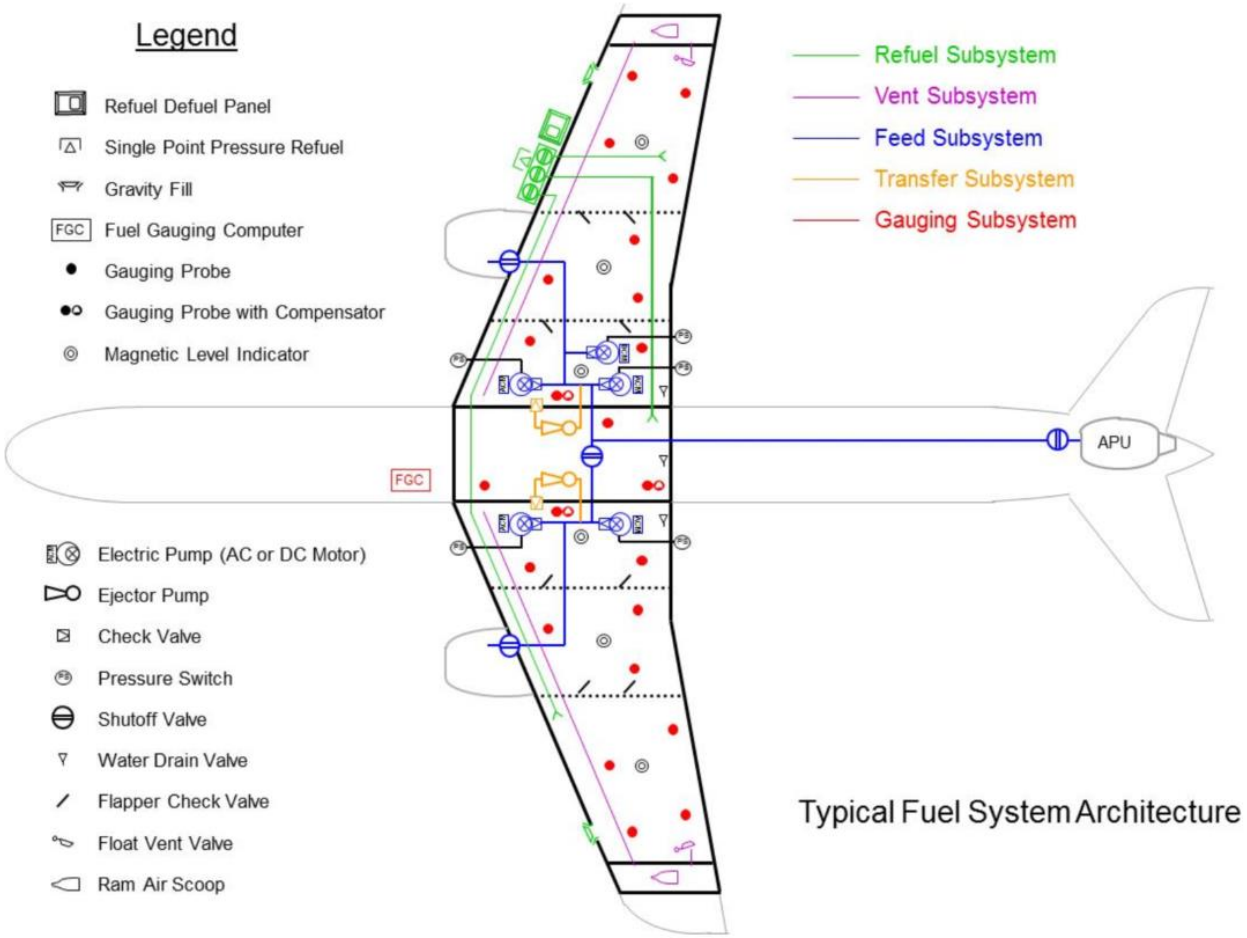

Figure 1 – Typical fuel system architecture [13]

Within the scope of the present paper a simplified fuel system model is used for geometry-based and knowledge-based engineering demonstration purposes. The simplified fuel system model includes the refuel subsystem and the feed subsystem since both have similar design requirements and approaches (see Subsection 4.3). The feed subsystem is further divided into the engine feed, APU feed and cross

feed. While the boost pumps are also included, any other components, such as valves, gauging probes etc., are not considered.

The outer aircraft geometry, similar to a B767, is used as a design space due to its availability on a conceptual level from the Avacon project [14]. The geometry data is stored in a CPACS format (Common Parametric Aircraft Configuration Schema) [35] and can be displayed in the TiGL Viewer [15]. In [3] the fuel system of a B777 is described in detail, which is assumed to be a comparable reference for the design of the present simplified fuel system model. Both, the design space and the simplified fuel system model are exemplary shown in Figure 2.

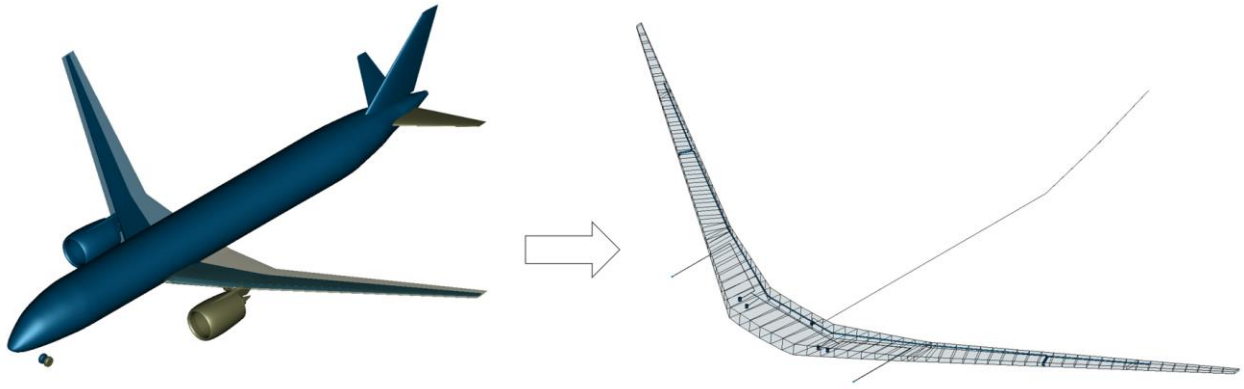

Figure 2 – Aircraft model from CPACS data file(left) and derived fuel system model within design space (right)

3. The Codex framework

The present approach is built on the COllaborative DEsign and eXploration (Codex) framework [2], which is a knowledge-based engineering (KBE) tool based on Semantic Web Technologies and is currently under development at the German Aerospace center (DLR). Codex is written in the programming language *Kotlin*, which allows the creation of custom *Domain-Specific Languages* (DSL) within the language. This framework enables the users to create domain-specific knowledge-bases and integrate these into a single model of the overall product. Codex consists of several modules, each of them providing its own DSL and ontology. The framework can easily be extended by creating additional plugin modules for specific tasks or applications. The core modules which are used here are the following:

- *codex-semantic* is the core module of the framework and is used for the creation and manipulation of semantic models. For this aim, it makes use of existing and well-established standards for knowledge representation such as RDF (Resource Description Framework) and OWL (Web Ontology Language) [16].
- *codex-rules* enables the definition of *production rules* [17]. The rules can be expressed via a custom DSL and allow for the dynamic, rule-based creation of system components.
- *codex-parametric* allows users to semantically define and solve systems of parametric constraints with a DSL that closely resembles common mathematical notation.

The *codex-geometry* module is also developed as part of the Codex framework. *codex-geometry* provides an ontology describing primitive geometric shapes, like spheres, cylinders, cuboids etc. Polylines and bezier curves can be modeled and used to create extrusions and pipes. Transformations on shapes and operations between different shapes (e.g. union, intersection etc.) allow the user to model complex geometries, following the principle of *Constructive Solid Geometry* [18]. For visualization purposes and enabling computations like the volume of a complex shape, codex-geometry uses the

OpenCascade CAD kernel [19] and the java bindings provided by JCAE [20]. Using these tools, the geometry can also be exported to BREP, STEP or STL files. Figure 3 shows the creation of an example shape with codex-geometry. Here the *GeometrySyntax* is used, which is the custom DSL that *codex-geometry* provides for creating all kinds of geometric objects and operations. Using this syntax, first several points are created. Each of this points corresponds to an individual in the semantic model. Next, some primitive geometric objects are created, which are also added as individuals to the semantic model: A cuboid, a sphere and three different cylinders. The final shape (which is a common CSG example) is then obtained by intersecting the cuboid with the sphere and then subtracting the union of the three cylinders from the shape, which can be efficiently written in the *GeometrySyntax* as `val result = (cub intersect sph) - (cyl1 + cyl2 + cyl3).`

The resulting shape is exported to a BREP file and displayed in the TiGL viewer [15].

Figure 3 – Creating an example shape with codex-geometry

Using the geometry ontology, geometric knowledge from different domains can seamlessly be integrated, even if the domain-specific models have no connection whatsoever. For example, for the fuel system use case a geometric representation is created from an existing wing design and then used as the design space for the creation of the fuel system (cf. Figure 2).

4. Methodology

The presented approach can be divided into the following steps:

- Create a domain-specific ontology describing the components of a fuel system
- Create the actual instances (supply lines, pumps etc.) for the example fuel system and their geometric representation
- Define geometric requirements and analysis functions for the evaluation of the example model, which are automatically evaluated

The following sections describe how the Codex framework is used for the three steps mentioned above.

4.1 Creating a Domain-Specific Ontology and Rules

In a semantic-web application, a model consists of *classes*, *individuals* (entities), and object- as well as data-*properties*. All knowledge in the model is expressed as *statements* in the form "[subject] [predicate] [object]" [21].

In order to create the example application, first a domain-specific ontology describing the concepts of the application is created. For this, the codex-semantic module is used. This ontology defines the

classes (e.g. *SupplyLine*, *FuelType*, *Pump*) and properties (e.g. *hasFuelType*, *hasMass*, *hasOuterRadius*) that are needed to describe the concrete design in the next step.

At this step, as much domain knowledge as possible is expressed in the ontology as well as in rules. The codex-parametric module is used to express known relationships between data properties as parametric equations. This includes very basic facts like the correlation between the radius and diameter of a pipe, but also domain-specific knowledge like Barlow's law for determining the required wall thickness of a pipe. Such relationships, which can be expressed as equations, are modeled as *parametric rules* in Codex. Figure 4 shows an example for such a rule, which applies Barlow's law to a linear pipe segment.

```
val barlowsLawLinear by Rule(FS, GKM) {  this: ParametricRuleBuilder
    When<SRDFResource, SRDFResource, SRDFResource, SRDFResource> {segment, supplyLine, fuelSystem, material ->
        +(segment..a..FS.LinearSegment)
        +(segment..FS.belongsTo..supplyLine)
        +(supplyLine..FS.belongsTo..fuelSystem)
        +(supplyLine..FS.hasMaterial..material)
    } Apply { segment, supplyLine, fuelSystem, material->
        eq( lhs: segment[FS.hasReqOuterRadiusTensile] - supplyLine[FS.hasInnerRadius],
            rhs: 2.0 * segment[FS.hasReqOuterRadiusTensile] * supplyLine[FS.hasInternalPressure] /
                (2.0 * material[FS.hasStrength] / fuelSystem[FS.hasSafetyFactor] + supplyLine[FS.hasInternalPressure]))
    }
}
```

Figure 4 – Example for a parametric rule

The codex-parametric engine is responsible for solving the resulting equation systems and will solve the equations as soon as enough parameters are available, regardless which parameters were provided by the user and which might be computed during the processing by some other rule. This gives the user a great deal of flexibility when creating a design.

Furthermore, the codex-rules module is used to formulate *production rules* describing how to "translate" the domain-specific objects into geometric shapes, i.e. creating the corresponding individuals in the codex-geometry system. Similar to the parametric rules, the production system will execute a production rule automatically as soon as all necessary knowledge is available. For example, a production rule can describe how to create a geometric representation of a pipe from a set of points and parameters like curve radii, wall thickness etc. This rule will be automatically executed for each individual of type *Pipe* when all required properties are available. Some of these properties probably have been set by the user at the start, others might be computed by other rules in the system, like the required wall thickness by the rule shown in Figure 4.

An interesting aspect of the creation of a three-dimensional pipe network is the geometrical calculation of a curved supply line segment. To facilitate the geometric integration, the spline of a circular arc is approximated by a cubic Bézier curve. To create the cubic Bézier curve, four control points are calculated. These support points can be derived imposing two constraints: First, the start and end point of the arc must be identical with the the start and end points of the Bézier curve, as well as their first derivatives. Second, the middle of the Bézier curve must lie on the circular arc. An example of a two-dimensional approximation of a circular arc by a Bézier curve and a more detailed explanation is given in [22].

This example focuses mainly on the shapes of the supplylines and the design space. The pumps are not modeled in detail, but their placement is addressed by allocating a cylinder-shaped space for each pump.

4.2 Creating the Example Fuel System

Based on the domain-specific ontology, an actual example design is built by creating individuals for the specific pipes, pumps etc. For each individual, input parameters can be specified by adding statements to the semantic model (e.g. *"refuelCrossline hasInnerRadius 0.04"*).

Similarly, the geometric requirements (discussed in Section 4.3) are added as statements to the model (e.g. *"refuelCrossline containdedIn designSpace"*).

The designer can also make use of the production rules here by introducing dependencies between supply lines. For example, if a connecting line should be created between $Line1$ and $Line2$, it is possible to choose a point $P1$ on $Line1$ as the starting point of the connecting line and define the endpoint $P2$ as "the closest point to $P1$ on $Line2$". Please note that at the time this statement is made, $P2$ can not immediately be computed as the shape of $Line2$ may not be generated, yet. However, there is a production rule for this "closest point" computation that will be automatically triggered once the $Line2$ shape is complete.

After adding all necessary statements, the parametric rules and the production rules are iteratively evaluated by Codex, thereby computing the missing parameters and generating the geometric shapes for all the individuals, until convergence is reached. Afterwards, the geometric requirements are evaluated in order to check if the resulting design is valid.

4.3 Defining Requirements and Analysis Functions for the Fuel System Model

The fuel system design is generally driven by different requirements and performance parameters like mass and pressure loss which are described subsequently. For the present fuel system model some exemplary requirements are defined. Technically, they are implemented as production rules, so they can be automatically evaluated by the codex-rules engine as soon as the respective geometric shapes are generated. In addition, two analysis functions are defined as parametric rules linking the structural mass as well as the fluid mechanical behavior to corresponding sizing parameters.

4.3.1 Requirements of the Fuel System Model

The fuel system model is evaluated with respect to several implemented requirements. The evaluation is based on geometric checks which link geometric components with each other. If any requirement is not fulfilled the Codex framework indicates the corresponding failed check. As shown in Table 1, four fundamental checks have been identified to formulate the requirements being applied to the fuel system model. However, additional checks and thus requirements can be easily implemented if needed.

Table 1 – Implemented checks and requirements

Check	Description	Requirements
containedIn	Check if volume A is entirely included in volume B	Refuel line, engine feed and cross feed shall be contained in the design space (outer wing geometry).
noOverlap	Check if volume A and volume B share any volume	All fuel lines shall not overlap with each other. The hazard of a uncontained rotor burst shall be minimized (explained below).
connectedTo	Check if volume A and volume B intersect with no gap	All fuel lines which have wanted intersections shall be able to exchange fuel (example below).
minimumDistance	Check if volume A and volume B have a spacing of the defined minimum distance or higher	Fuel lines shall have a minimum distance to the structure of 0.25 inches to surrounding structure (requirement not implemented yet due to missing detailed structure).

Uncontained rotor burst example
When integrating the fuel system into the design space, the hazards of an uncontained rotor burst shall be minimized. Three different cases of engine rotor failure are defined in AC20-128A [23] with respect to spread angle, fragment size and considered energy as shown in Table 2.

Table 2 – Uncontained engine rotor failure impact zones

Spread angle	Fragment size	Energy
±3°	1/3rd disk fragment	infinite energy
±5°	intermediate fragment	finite energy
±15°	small fragments	protection by primary structure

Following the recommendations of [23], the minimization of engine burst hazards can be achieved by relocating the components outside the debris impact zones, duplicate and separate critical components or protecting critical components using the airframe structure or additional shielding. This requirement is implemented by creating a geometric representation of the debris impact zone and then adding *noOverlap* requirements for all supply lines to this shape. According to [13] the 5° spread angle is usually the design driver, because it is difficult to justify shielding for the spread angle between 3° to 5°.

Connected fuel line example
In case of a preliminary design the *connectedTo* check detects gaps between intersecting fuel lines. Figure 5 shows an example with very small gaps which could be resolved after noticing the failed check.

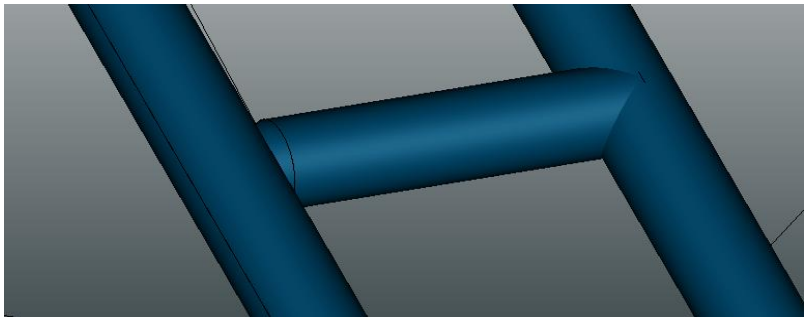

Figure 5 – Design problem detected by a *connectedTo* check

4.3.2 Structural Mass of the Fuel System Model

The structure of each fuel line is sized for a required internal and external maximum operating pressure as shown in Table 3. The main goal is to determine the wall thickness and resulting mass. The mass approximation does not consider additional structures such as fittings or clamping. Consequently, the following approach rather results in a mass trend depending on varying sizing parameters than in a complete fuel system mass model.

Table 3 – Maximum operating pressure for fuel lines according to SAE ARP 8615 [24]

Fluid subsystem	Internal pressure (psig)	External pressure (psig)
Engine feed	60	15
APU feed	60	15
Crossfeed	60	-
Refueling line	90	3

Depending on the location and application there are rigid and flexible fuel lines. Rigid stainless steel fuel lines are used at areas which are subject to damage or heat such as the engine compartment.

Flexible fuel lines are typically made of a synthetic rubber interior reinforced by a fiber braid warp and a synthetic cover. They are used in areas with vibration between components such as engine and pylon. However, rigid fuel lines made of 5052 aluminum alloy are commonly used at remaining fuel system areas (being considered for the entire present fuel system model for simplification reasons). Table 4 shows the mechanical properties which are applied to the fuel system model. [25,26]

Table 4 – Mechanical Properties of 5052 aluminum alloy used as fuel line material [24,26,27]

Property	Value
Density	2.67 $\frac{g}{cm^3}$
Poisson's ratio	0.33
Elastic modulus for compression	70.7 GPa
Maximum stress @ 10^5 Cycles	130.0 MPa

The wall thickness is sized for the following three different failure modes according to AD2000 [28]:

- Tensile strength (under internal pressure)
- Compressive strength (under external pressure)
- Elastic buckling (under external pressure)

In addition, minimum thickness requirements for fuel lines are given by the SAE in [29]. The maximum thickness of the three failure modes and the minimum thickness requirement (=envelope) are considered to be a feasible wall thickness as exemplary shown in Figure 6. Potential deviation due to standardization of tube sizes are not taken into account.

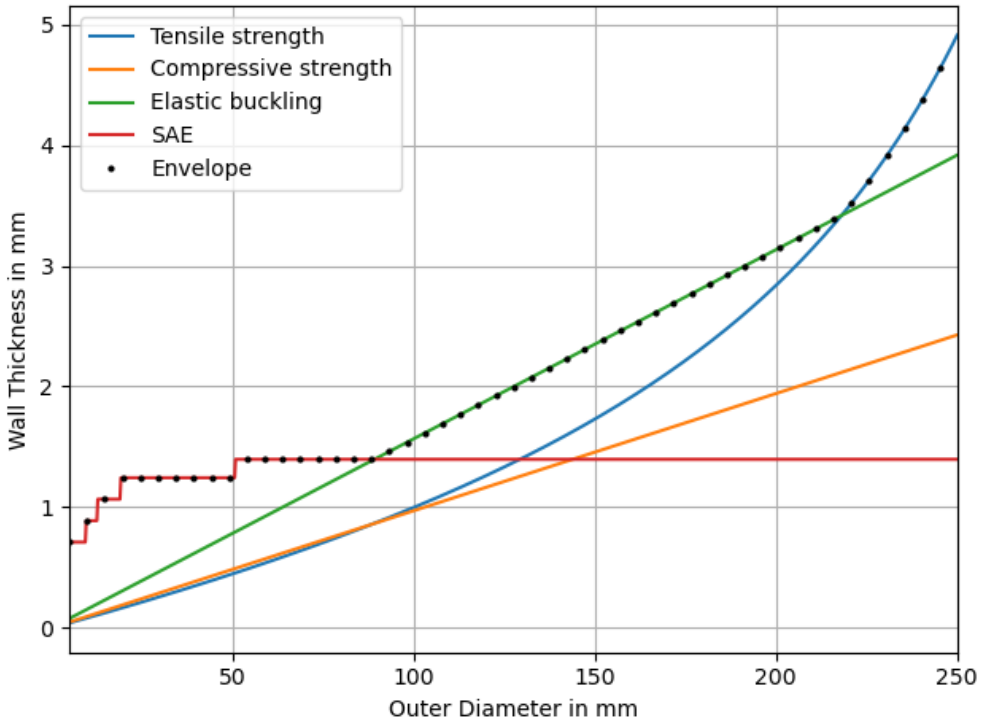

Figure 6 – Wall thickness depending on outer diameter based on refuel line conditions

4.3.3 Pressure Loss of the Fuel System Model

The pressure loss of the present fuel system model is only investigated with respect to the fuel flow in fuel lines. The fuel flow depends on the respective fuel line e.g. due to a time requirement for refueling or continuous fuel flow for the engine. Table 5 shows the required fuel flows for the present fuel system model.

Table 5 – Maximum fuel flow for reference aircraft according to B767's Airplane Characteristics for Airport Planning and ICAO Engine Exhaust Emissions Data Sheet

Fluid subsystem	Fuel flow (kg/s)
Engine feed	2.083
APU feed	0.03
Crossfeed	2.083
Refueling line	26.87

Commonly, Jet-A or Jet-A1 is used as commercial aviation fuel [13]. Due to its complex chemistry, the fuel properties may differ within certain limits. The specification limits and the implications of fuel system design with regard to different fuel properties are described in [30]. For the pressure loss analysis of the fuel system model the density and viscosity are necessary fuel related parameters. The density changes linearly with temperature within the temperature range of aircraft operation [30]. A temperature deviation of 100 K results in a density change of about 10 % for a representative measured fuel sample [31]. The viscosity varies considerably with temperature: From [31] the same fuel sample showed an increase from 500 μPas to 2750 μPas by lowering the temperature from 370 K to 270 K. To simplify the analysis function a reference temperature of 290 K is chosen and the resulting values for density and viscosity are 806 kg/m^3 and 1550 μPas, respectively. The determination of pressure losses within the pipe system is conducted for linear segments and bends. In [32] the pressure loss of a fluid flow in a circular pipe due to friction is described and shown in Equation (1).

$$\Delta p = f \frac{l}{d_i} \cdot \frac{\rho V^2}{2} \tag{1}$$

In this regard f is the friction factor, l the length of the segment, d_i the inner diameter, ρ the density and V the fluid velocity. The friction factor f depends on the Reynolds number Re and the relative roughness of the pipe wall. Typical values for aircraft type tubing and the corresponding friction factor are provided by [33]. These values are approximated by the empirical correlations for laminar and turbulent flows up to $\mathrm{Re} < 10^6$ given in [32] for a smooth pipe. For higher Reynolds numbers the impact of the wall roughness is taken into account by interpolating the friction factor based on aircraft type tubing in [33]. The total pressure loss of bends includes the friction losses and an additional bend depending loss as stated in [33] and shown in Equation (2).

$$\Delta p = \left[f \frac{l}{d_i} + C \cdot K_{t90} \right] \cdot \frac{\rho V^2}{2} \tag{2}$$

The loss coefficient due to 90° bends K_{t90} depends on the ratio of inner diameter and bend radius as well as the friction factor. The correction factor C for bends other than 90° is interpolated from a chart.

5. Demonstration by a Parametric Study

The capabilities of the Codex framework are demonstrated by a parametric study. The latter considers all requirements and analysis functions of Subsection 4.3 for defined geometric input parameters. As

mentioned before, it is not claimed to show a complete design study for a fuel system but an exemplary parametric study being related to realistic design trades.

5.1 Input Parameters

Each fuel line of the fuel system model is described by the inner pipe radius, fuel line length and bend radius (the latter in case of a curved segment). The length is a result from the built fuel system model as described in Section 2 and Section 4. The bend radius is set to two times the inner pipe radius. The inner pipe radius is used as the main input variable for the parametric study. As stated in Equation (3) it depends on the fuel flow $\dot{m}$, fuel velocity v and fuel density ρ.

$$\dot{m} = \pi r_i^2 \cdot v \cdot \rho \tag{3}$$

The inner radius is derived from a fuel velocity range while the other parameters are defined above. According to [3] the fuel velocity in feed lines is kept below approximately 3 m/s. However, the fuel velocity in the APU feed is even further reduced due to its low fuel flow requirement (otherwise very small, not realistic inner radii may result).

For the refuel line the opposite case is true: Very high fuel flow requirements lead to high radii and/or high fuel velocities. Nevertheless, the maximum combination of inner pipe radius and fuel velocity is limited by the static discharge requirement. The fuel flow through pipes generates static electricity due to friction. Consequently, a discharge may cause ignition if explosive mixtures are present [34]. According to [34] the limitation of Jet A-1 is shown in Equation (4).

$$v \cdot r_i \leq 0.25\ m^2/s \tag{4}$$

Taking the considerations from above into account results in the value ranges as shown in Table 6. Each row represents one sizing case being evaluated in the following subsection.

Table 6 – Fuel velocity range and resulting inner pipe radius range for parametric study

Sizing case	Engine feed		APU feed		Crossfeed		Refuel line	
	v (m/s)	r_i (m)	v (m/s)	r_i (m)	v (m/s)	r_i (m)	v (m/s)	r_i (m)
1	1.524	0.023	0.1524	0.0088	1.524	0.023	4.572	0.048
2	1.8288	0.021	0.18288	0.0080	1.8288	0.021	4.8768	0.047
3	2.1336	0.020	0.21336	0.0075	2.1336	0.020	5.1816	0.045
4	2.4384	0.018	0.24384	0.0070	2.4384	0.018	5.4864	0.044
5	2.7432	0.017	0.27432	0.0066	2.7432	0.017	5.7912	0.043
6	3.048	0.016	0.3048	0.0062	3.048	0.016	6.096	0.042

5.2 Results

The results of the parametric study are shown in Figure 7. The mass trend, as explained in 4.3.2, is shown against the cumulative pressure loss of the pipe network. Based on the implemented analysis functions higher pipe radii result in higher mass but in lower pressure loss and the other way around. Hence, a trade-off between mass and pressure loss should be considered when designing fuel systems. However, a potential optimization has to be conducted on aircraft level with both penalties (mass and pressure loss) being combined in one assessment parameter e.g. the block fuel of a representative mission. Such an optimization trade is out of this study's scope.

In addition, four out of six sizing cases result in an overlap of two pipes which are marked as failed sizing cases. The information can be used to sort out the failed sizing cases or to change the fuel line way points for instance.

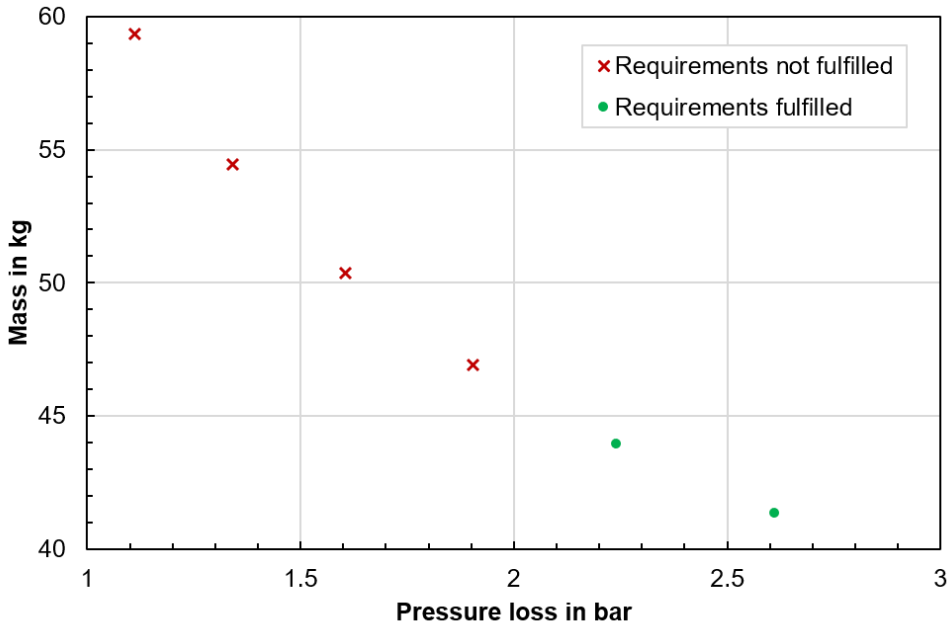

Figure 7 – Results of parametric study (decreasing radius from left to right)

Figure 8 shows the uncontained rotor disk failure requirement for the 5° spread angle case and for one representitive sizing case from Table 6. All components or fuel lines being located within this impact zone can be checked and based on the results the integration can be adapted in order to minimize the hazard for this failure case.

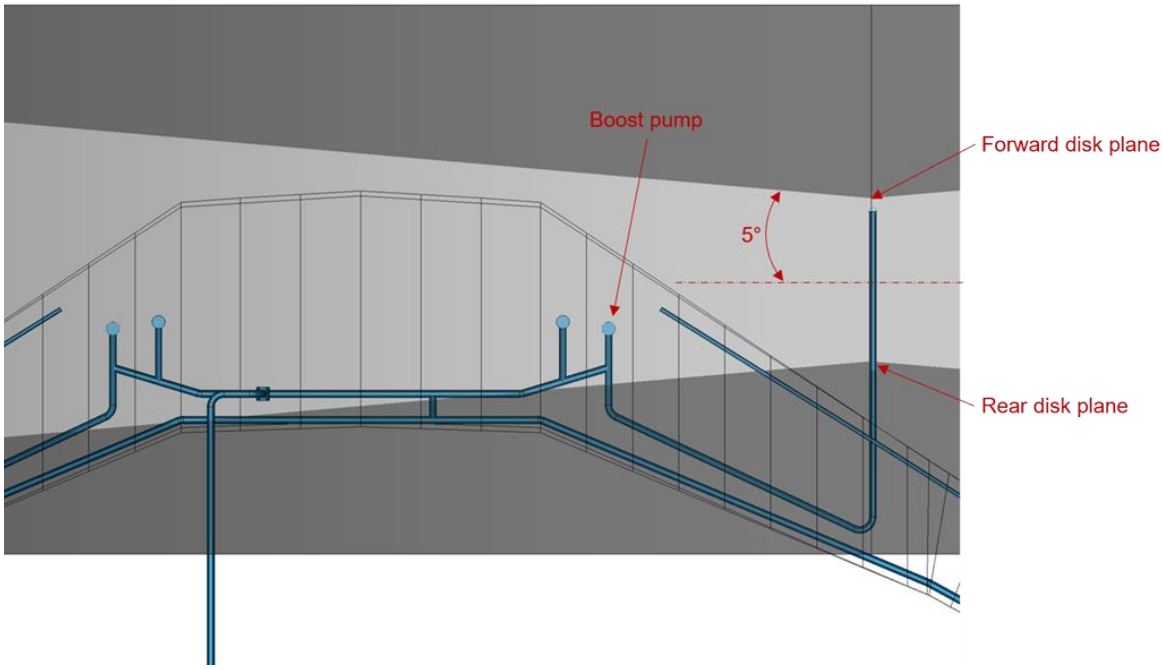

Figure 8 – Example for uncontained rotor disk failure for 5° spread angle

6. Conclusion and Outlook

This paper presented an approach to address the geometric integration of subsystems at an early design stage of the aircraft design process. It uses and extends the Codex framework which is based on Semantic Web technologies. The capabilities of this approach were demonstrated based on the example use case of an aircraft fuel system. Geometric requirements as well as analysis functions were modeled to evaluate the resulting example sizing cases.

In conclusion, the combination of semantic modeling, design requirements and analysis functions can support the aircraft design process. In some cases, simple geometrical checks can be used to model complex design requirements. An important prerequisite for many checks is having a geometric representation of the design space. Overall, the usage of the Codex framework makes the system flexible and easy to extend. Plans for future work include improving the level of detail of the fuel system model, implementing a wider range of requirements and enhancing the analysis functions. In addition, the approach may be applied to further use cases.

In this paper, the requirements and analysis functions were used to evaluate and compare a number of different fuel system sizing cases in a parametric study. In the bigger picture, this is the first step towards an automated design of geometric subsystems consisting of synthesis and analysis. In their future work the authors plan to approach the automatic generation of a geometric design based on user-defined requirements and target functions.

7. Contact Author Email Address

brigitte.boden@dlr.de

8. Copyright Statement

References

[1] Raymer, D (2012). *Aircraft Design: A Conceptual Approach*. American Institute of Aeronautics and Astronautics.

[2] Zamboni, J, Zamfir, A and Moerland, E (2020). Semantic knowledge-based-engineering: The codex framework. *12th international joint conference on knowledge discovery, knowledge engineering and knowledge management*. **2** 242–9

[3] Langton, R, Clark, C, Hewitt, M and Richards, L (2009). *Aircraft Fuel Systems*. John Wiley & Sons.

[4] TXT Group Pacelab APD. https://pace.txtgroup.com/products/preliminary-design/pacelab-apd/

[5] ParaPy B.V. ParaPy - Knowledge Based Engineering Platform. https://www.parapy.nl/

[6] Johannes Groß (2013). *Aufbau Und Einsatz von Entwurfssprachen Zur Auslegung von Satelliten*. Dissertation. Universität Stuttgart, Stuttgart

[7] Schaefer, J and Rudolph, S (2005). Satellite design by design grammars. *Aerospace Science and Technology*. **9** 81–91

[8] Schmidt, J (2017). Total Engineering Automation: Vision and Realization with Graph-based Design Languages and the Design Cockpit 43 Software suite. https://www.iils.de/

[9] Vogel, S and Rudolph, S (2016). Automated piping with standardized bends in complex systems design. *International Conference on Complex Systems Design & Management*. 113–24

[10] Burschyk, T, Cabac, Y, Silberhorn, D, Boden, B and Nagel, B (2021). Liquid hydrogen storage design trades for a short-range aircraft concept. *Deutscher Luft- und Raumfahrtkongress 2021*

[11] Moir, I and Seabridge, A (2008). *Aircraft Systems*, 3. ed. John Wiley & Sons, Chichester

[12] FAA (2016). *Pilot's Handbook of Aeronautical Knowledge*. Oklahoma City

[13] SAE International (2019). Aircraft fuel system design guidelines (AIR7975)

[14] Woehler, S, Hartmann, J, Prenzel, E and Kwik, H (2018). Preliminary aircraft design for a midrange reference aircraft featuring advanced technologies as part of the avacon project for an entry into service in 2028. *Deutscher Luft- und Raumfahrtkongress*. Friedrichshafen

[15] Siggel, M, Kleinert, J, Stollenwerk, T and Maierl, R (2019). TiGL: An open source computational geometry library for parametric aircraft design. *Mathematics in Computer Science*. Springer. **13** 367–89

[16] Semantic Web. https://www.w3.org/standards/semanticweb/

[17] Norvig, P and Russell, S (2016). *Artificial Intelligence: A Modern Approach, Global Edition*. Pearson Education Limited, 1152p

[18] Foley, J D, Van, F D, van Dam, A, Feiner, S K, Hughes, J F and Hughes, J (2010). *Computer Graphics: Principles and Practice*, 2. ed. in C, reprinted with corr., 25. print. Addison-Wesley, Boston

[19] OpenCascade. https://www.opencascade.com/

[20] JCAE. http://jcae.sourceforge.net/

[21] Allemang, D and Hendler, J A (2012). *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*, 2. ed. Elsevier, Morgan Kaufmann.

[22] Riškus, A (2006). Approximation of a cubic bezier curve by circular arcs and vice versa. *Information Technology and Control*. **35**

[23] FAA (1997). Design considerations for minimizing hazards caused by uncontained turbine engine and auxiliary power unit rotor: AC 20-128A. https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_20-128A.pdf

[24] SAE International (2011). Fuel system components: General specification for (ARP8615)

[25] FAA (2018). *Aviation Maintenancettechnician Handbook Airframe*. Oklahoma City

[26] ASM International (1990). *Properties and Selection: Nonferrous Alloys and Special-Purpose Materials*

[27] ASM International (1996). *Fatigue and Fracture*

[28] Verband der TÜF e.V. (2016). *AD2000 Regelwerk: Taschenbuch 2016*. Berlin

[29] SAE International (2013). Fuel and oil lines, aircraft, installation of (AS18802)

[30] SAE International (2018). AE-5A Aerospace Fuel, Inerting and Lubrication Sys Committee Guidance on the impact of fuel properties on fuel system design and operation. SAE International, 400 Commonwealth Drive, Warrendale, PA, United States

[31] Huber, M L, Lemmon, E W and Bruno, T J (2010). Surrogate mixture models for the thermophysical properties of aviation fuel jet-a. *Energy & Fuels*. **24** 3565–71

[32] VDI (2013). *VDI-Wärmeatlas: Mit 320 Tabellen*, 11., bearb. und erw. Aufl. Springer Vieweg, Berlin

[33] CRC - Coordinating Research Council (1962). Aviation handbook: Fuels and fuel systems

[34] American Petroleum Institute (1998). Protection against ignitions arising out of static, lightning, and stray currents. *API RP 2003 6TH ED*

[35] Alder M, Moerland E, Jepsen J and Nagel B (2020). Recent Advances in Establishing a Common Language for Aircraft Design with CPACS. *Aerospace Europe Conference 2020*, Bordeaux, France