34th Congress of the International Council of the Aeronautical Sciences
Florence
September 9-13
ICAS
CONGRESS
2024
ITALY

AUTOMATING THE VERIFICATION OF GEOMETRIC REQUIREMENTS FOR AIRCRAFT FUEL SYSTEMS USING KNOWLEDGE-BASED ENGINEERING

Brigitte Boden[1], Matheus Padilha[1], Tim Burschyk[1], Carlos Cabaleiro de la Hoz[1], Erwin Moerland[1] & Marco Fioriti[2]

[1]Institute of System Architectures in Aeronautics, DLR, Hein-Saß-Weg 22, 21129 Hamburg
[2]Department of Mechanical and Aerospace Engineering, Politecnico di Torino,
C.so Duca degli Abruzzi 24, 10129 Turin

Abstract

The definition of and compliance to requirements is one of the most important steps of the aircraft design process. For the aircraft fuel system, many rules and requirements exist in certification specifications and guidelines. As the fuel system is highly integrated with multiple other aircraft subsystems, its design is highly impacted by the geometric definition of the aircraft and the positioning of other subsystems. Therefore, it is important to carefully consider geometric requirements already in early design stages.

Although being a crucial step in the design process, checking all the requirements can currently be a difficult, tedious and time-consuming task for aircraft designers. This is especially the case in early design stages, since the designs generally change considerably and often. Therefore, this work aims to define a methodology to support aircraft fuel system design while taking geometric, functional, and certification constraints into account.

This paper presents a rule-based approach for the verification of geometric and certification requirements. We use the Codex (COllaborative DEsign and eXploration) platform, a modern framework for the creation of knowledge-based engineering tools, to model critical components and requirements for the fuel system and to automatically check the compliance with those requirements. Next to having the potential to significantly speed-up the design process, utilizing such automatic verification procedures reduces the burden of having to perform manual and repetitive tasks. By ensuring compliance with the requirements it can also reduce the amount of rework in later stages of the design process.

Keywords: Fuel System, System Installation, Interface Requirement, Knowledge-Based Engineering, Certification

1. Introduction

Aircraft fuel systems are essential on-board systems. Their task is to store fuel and to safely and efficiently deliver it to the engines, allowing the propulsion system to generate the necessary thrust for the flight of the aircraft. They have to be designed to store and distribute fuel throughout the aircraft, ensuring a constant and reliable supply during flight. Thus, the fuel system design is a top priority in aircraft engineering.

The fuel system must be carefully engineered to account for factors such as fuel weight, balance and distribution. Furthermore, stringent safety regulations and protocols must be followed for the certification of the aircraft [1] [2].

As the fuel storage requires much volume and the fuel system is highly integrated with multiple other systems of the aircraft, its complex geometry impacts the overall performance and efficiency of the system as well as the safety and reliability of the aircraft. The fuel system design is heavily impacted by its geometric definition, which makes it important to carefully consider its geometry already in the early stages of development. Accurate geometric knowledge about this system is also crucial for effective integration with other subsystems of the aircraft. This work aims to define a methodology to support aircraft fuel system design while taking geometric, functional, and certification aspects into account.

Knowledge-based engineering (KBE) approaches are used to model knowledge in a structured way when working with multidisciplinary designs. They can provide highly specialized environments and languages which are fit to their respective domain of application. The use of Semantic Web Technologies (SWT) [3] delivers a domain-neutral way of knowledge formalization as well as an easy and efficient integration of knowledge between different domains.

The COllaborative DEsign and eXploration (Codex) platform [4] [5], which is developed at the German Aerospace center (DLR), aims at combining KBE and SWT technologies to support the digitalization of the aircraft design process. The framework targets the creation of domain-specific knowledge-bases and enables integrating these into a single model of the overall product. Thereby, engineering knowledge as well as design guidelines and requirements can be simultaneously incorporated to provide aircraft designers the capability to evaluate their design decisions and to deduce additional suggestions for even more efficient designs. Particularly for geometric definitions, the application of the Codex platform has the potential to enable designers to concentrate on the creative part of their design tasks, while being assisted by automated consistency checking.

The *codex-geometry* module [6], which is part of the Codex framework, provides an ontology describing primitive geometric shapes (like spheres, cylinder, curves etc.), transformations on shapes and operations between different shapes (e.g. union, intersection etc.), which in combination allow the user to model complex geometries. Furthermore, it allows the user to formulate geometric requirements, which can be automatically evaluated by Codex using a declarative rule-based approach. *Codex-geometry* was first introduced in [6], which showed a very preliminary version of the fuel system model as an example use case. In contrast, our current model is much more detailed and takes additional certification requirements into account.

In this work, we use *codex-geometry* for creating a detailed geometric model of the fuel system as well as geometric requirements derived from certification specifications and guidelines. The Codex rule system is used to automatically check the compliance of the geometry with the given requirements. The reference aircraft used to test our approach is an A320 like configuration, created at the German Aerospace Center and is referred to as D150 [7].
As use case, the effect of a main rotor burst and its potential to affect critical systems is studied. In this work, we developed a tool to estimate the probability for certain components to be hit in case of a rotor burst.

1.1 Related Work

Knowledge-Based Engineering systems provide specialized environments and languages for their specific domain of applications. For the aerospace sector, two examples of KBE tools are Pacelab APD [8] and ParaPy [9], both of which use a hierarchical model for modelling the data and an object-oriented programming language for the creation.

Schäfer et al. [10] discuss the creation of “design grammars” or graph-based design languages for the satellite design process. Design Cockpit 43 [11] uses graph-based design languages to store engineering knowledge in order to automate the design process. In [12], an algorithm for the automated generation of pipe routes in a given installation space is proposed, with is also based on a design language. These design languages are based on the Unified Modeling language (UML) and use an object-oriented approach.

In contrast, the Codex framework [4] uses a non-hierarchical approach based on Semantic Web Technologies. In contrast to hierarchical approaches, this approach is better suited to easily integrate knowledge from different disciplines and to create a highly collaborative KBE application. Based on this framework, the Codex platform [5] is being developed as a web-based collaborative engineering platform.

In [6] the *codex-geometry* module has been introduced and demonstrated on a preliminary version of our fuel system model. *codex-geometry* has also been used for modelling a liquid hydrogen tank and performing volume computations [13]. [14] describes how to use Codex for the rapid generation of novel air vehicle configurations for combat aircraft design.

1.2 Structure of this paper

This paper is structured as follows: Section 2 introduces the Codex platform and discusses how the example fuel system is created within the platform. Section 3 describes several key geometric requirements for the fuel system and how they can be verified using the *codex-geometry* module. Section 4 introduces a tool for checking the impact of an uncontained rotor burst based on *codex-geometry*. Section 5 provides a short outlook on the next steps planned for future work.

2. The Codex platform

The COllaborative DEsign and eXploration (Codex) platform [4] is a tool for the development of knowledge-based engineering (KBE) applications based on Semantic Web Technologies. The first version of Codex is a framework which allows the user to create custom *Domain-Specific Languages* (DSL) for their specific domains. Based on these languages, users can create domain-specific knowledge bases and easily integrate these into a single model of the overall product. The framework consists of different modules, the most important being:

codex-semantic is the core module of the framework. It is used for the creation and manipulation of semantic models, using standards for knowledge representation such as the Resource Description Framework (RDF) and the Web Ontology Language (OWL).

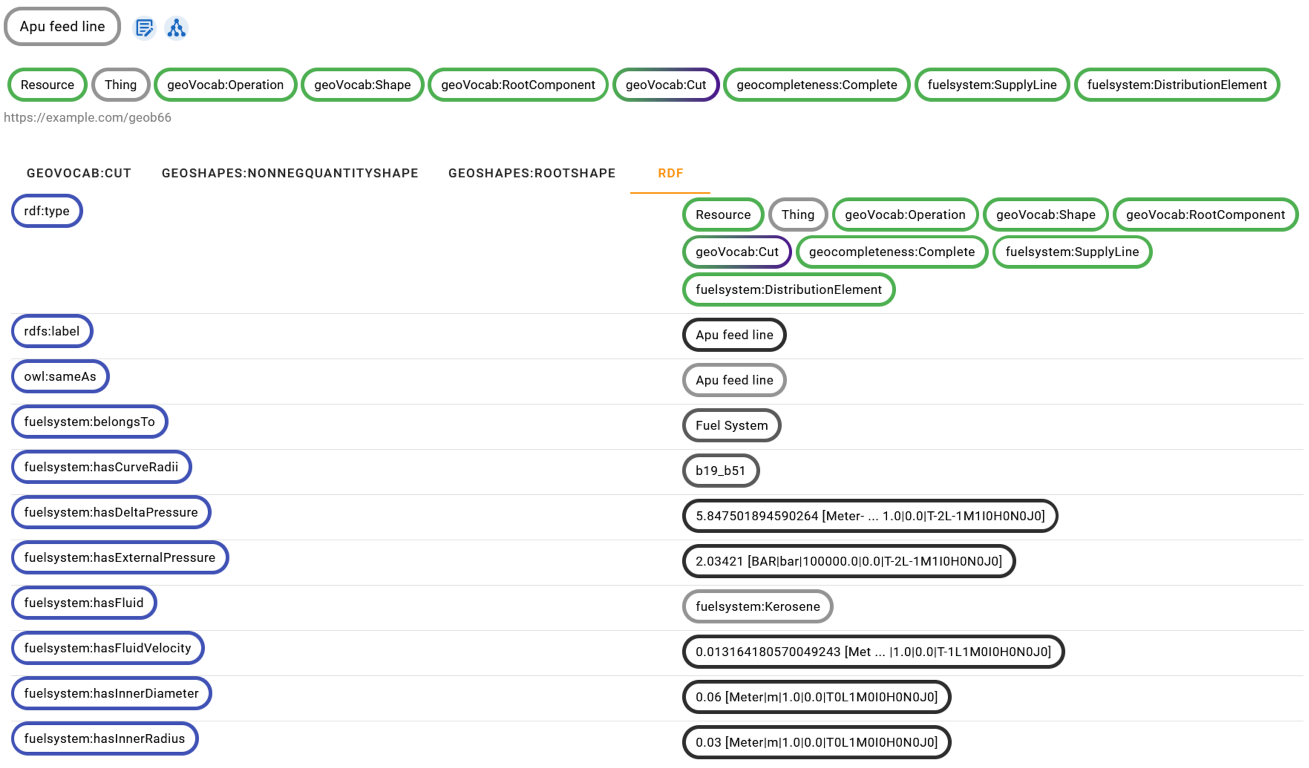
codex-rules enables the definition of so-called production rules. The rules can be expressed via a custom DSL and allow for the dynamic, rule-based creation of system components. Each rule consists of a Left-Hand-Side, which is basically a query on the knowledge-graph similar to SPARQL SELECT, and a Right-Hand-Side, which contains code that will be executed for every match in the corresponding model.

codex-parametric allows users to define systems of mathematical constraints which can are evaluated by a solver. Therefore, it provides a special DSL that closely resembles common mathematical notation.

codex-geometry [6] adds capabilities for geometric modelling and provides its own DSL which allows users to create the geometric parts of their model efficiently, as well as the formulation of geometric requirements. It uses the Open Cascade CAD kernel [15] in the background for generating geometries for visualization and for computations like the volume of a complex shape.

Using this first version of the Codex framework requires a certain level of programming skills and a basic understanding of SWT. To make usage of the framework even easier and more accessible, currently a web application is being developed on top of the existing framework [5].

Figure 1 shows an example of an individual representing the Auxiliary Power Unit (APU) feed line in the RDF view of the Codex web application. At the top right all the classes for this individual are listed. Please note that the individual has different classes from different domains, which would not be possible in a conventional object-oriented approach: From the point of view of the geometry domain, it is a Cut, which is a certain type of shape. From the point of view of the fuel system, it represents a Supply Line. Below, the properties of the individual are listed, which also come from different domains.


Apu feed line

Resource · Thing · geoVocab:Operation · geoVocab:Shape · geoVocab:RootComponent · geoVocab:Cut · geocompleteness:Complete · fuelsystem:SupplyLine · fuelsystem:DistributionElement

https://example.com/geob66

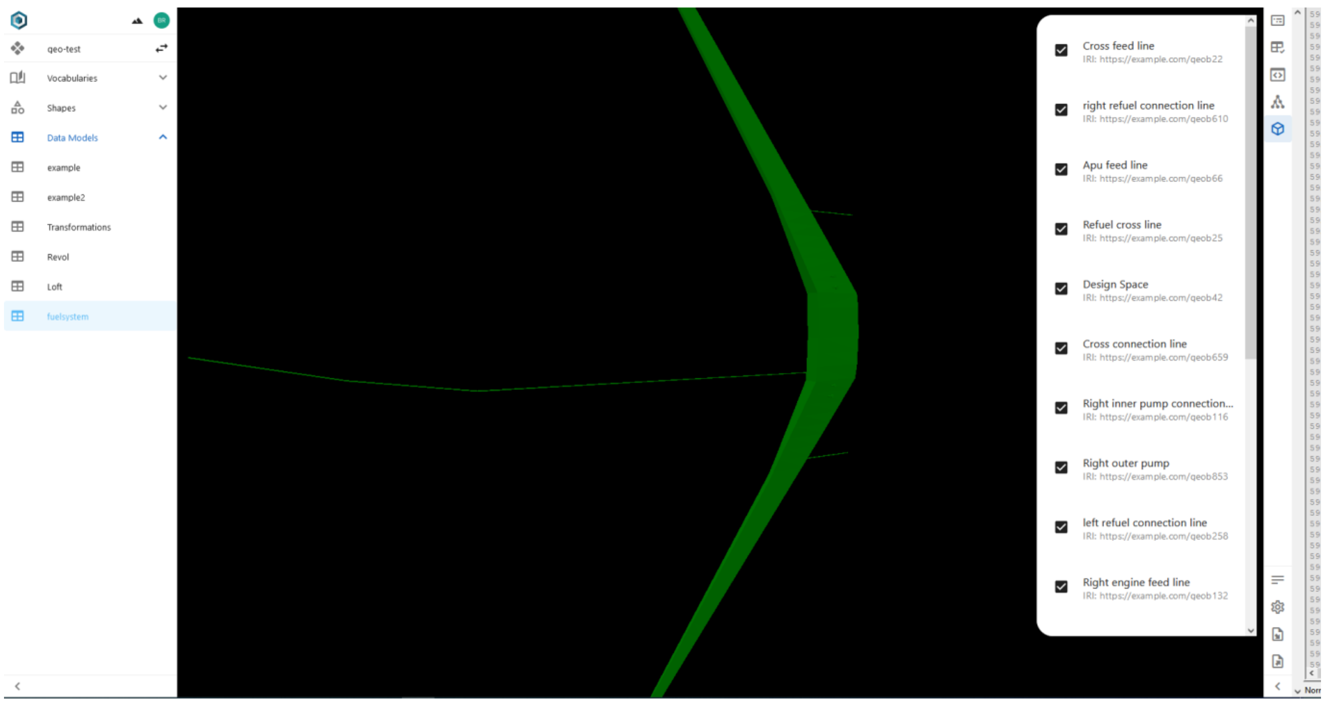
GEOVOCAB:CUT · GEOSHAPES:NONNEGQUANTITYSHAPE · GEOSHAPES:ROOTSHAPE · RDF

rdf:type	Resource, Thing, geoVocab:Operation, geoVocab:Shape, geoVocab:RootComponent, geoVocab:Cut, geocompleteness:Complete, fuelsystem:SupplyLine, fuelsystem:DistributionElement
rdfs:label	Apu feed line
owl:sameAs	Apu feed line
fuelsystem:belongsTo	Fuel System
fuelsystem:hasCurveRadii	b19_b51
fuelsystem:hasDeltaPressure	5.847501894590264 [Meter- ... 1.0|0.0|T-2L-1M1I0H0N0J0]
fuelsystem:hasExternalPressure	2.03421 [BAR|bar|100000.0|0.0|T-2L-1M1I0H0N0J0]
fuelsystem:hasFluid	fuelsystem:Kerosene
fuelsystem:hasFluidVelocity	0.013164180570049243 [Met ... |1.0|0.0|T-1L1M0I0H0N0J0]
fuelsystem:hasInnerDiameter	0.06 [Meter|m|1.0|0.0|T0L1M0I0H0N0J0]
fuelsystem:hasInnerRadius	0.03 [Meter|m|1.0|0.0|T0L1M0I0H0N0J0]

Figure 1 – An example individual (APU feed line) in the Codex web application

Besides the geometry DSL and rule set, *codex-geometry* also provides a web service which receives an RDF model containing geometric shapes and can return the generated geometry for displaying purposes or can answer requests like “What is the volume of shape x?”

The geometry view of the Codex web application (shown in Figure 2) uses the geometry service to generate the geometric shapes for all the components contained in the model and allows the user to select which components to display. Clicking on a component opens a window in which all stored information about the component can be browsed and changed if necessary, like shown in Figure 1.



- ✓ Cross feed line
  IRI: https://example.com/qeob22
- ✓ right refuel connection line
  IRI: https://example.com/qeob610
- ✓ Apu feed line
  IRI: https://example.com/qeob66
- ✓ Refuel cross line
  IRI: https://example.com/qeob25
- ✓ Design Space
  IRI: https://example.com/qeob42
- ✓ Cross connection line
  IRI: https://example.com/qeob659
- ✓ Right inner pump connection...
  IRI: https://example.com/qeob116
- ✓ Right outer pump
  IRI: https://example.com/qeob853
- ✓ left refuel connection line
  IRI: https://example.com/qeob258
- ✓ Right engine feed line
  IRI: https://example.com/qeob132

Figure 2 - Geometry View

The basic methodology to build a domain-specific application with *codex-geometry* was described in [6] and consists of three steps:

1. Create a domain-specific ontology providing the vocabulary for the specific domain (Here: Fuel System) as well as domain-specific rules
2. Create the actual instances (supply lines, pumps etc.) for the example fuel system and their geometric representation
3. Define geometric requirements for the example system, which are automatically evaluated by *codex-geometry*. The requirements are implemented using the production rule system offered by *codex-rules*.

In our current work, we also follow this methodology and focus on the creation of geometric requirements for the domain of the fuel system. In the following section, these requirements will be described in detail.

3. Key geometric requirements for the fuel system

For the design of modern aircraft, many airworthiness requirements are defined as certification specifications in EASA's CS-25 rules [1]. Others are specified in SAE design guidelines like [2], [16], or [17]. Checking aircraft designs for adherence to these requirements is a crucial step, thus automating it can highly support the design process. In other cases, requirements can be vague or not very precise, they might be derived from experience for a specific system and application case. The methodology shown here is completely open to new requirements, as users can easily add additional rules and checks.

As a reference aircraft to test our approach, we use the A320-like D150 [7] configuration created at the German Aerospace Center. The configuration is given as a CPACS (Common Parametric Aircraft Configuration Schema) [18] file and can be displayed in the TiGL viewer [19].

Using the *codex-geometry* module, 25 geometric requirements were modeled as rules that can be set by the user and automatically checked by the Codex system, including the following:

Center of gravity shift

Due to the large amount of fuel stored in the tanks and the complex geometry of the fuel system, it has a significant impact on the center of gravity of the aircraft. Thus, this verification function uses *codex-geometry* to generate the overall geometry of the fuel system, compute its center of gravity and check if it is within user-specified bounds.

Intersections between components

The “intersections between components” requirement is a check for errors in the design that cause components to intersect. It can be set for any pair of components and it verified by generating both shapes and computing the intersection between them using OpenCascade [15] methods.

The following figures show two examples of design problems that can be detected by this check. Figure 3 illustrates the intersection between a fuel pump and one of the wing’s ribs. The intersecting rib is displayed in red. A second example of intersecting components is illustrated in Figure 4, where the intersection of sections of fuel lines and fuel valves with the fuel tank’s physical boundaries is detected by the rules modeled in the Codex framework. In such cases, additional information is provided by the tool to allow corrective measures to correct the design. The non-compliant components are marked in red.

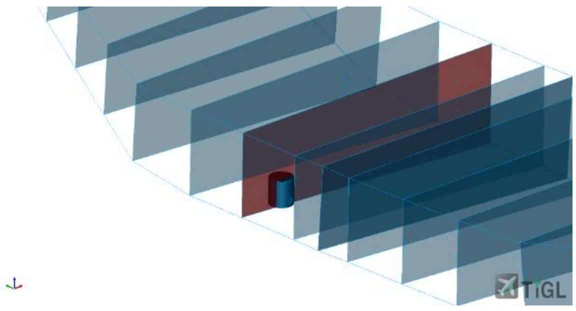


Figure 3 - Identified overlap between a rib and a fuel pump

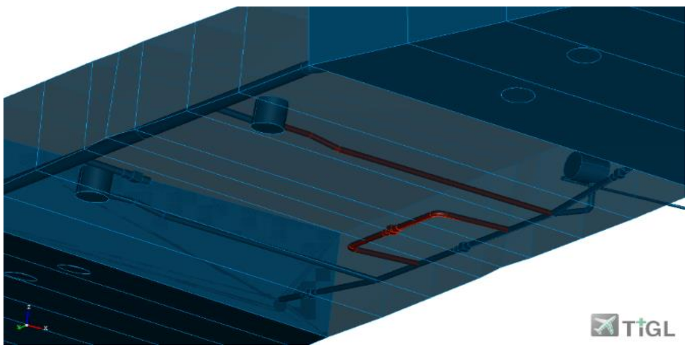


Figure 4 -- Example of a geometric design flaw due to an overlap between fuel system components and the system's physical boundaries

Minimum distance between lines and fuel tanks

Following design guidelines, certain minimum distances should be maintained between fuel lines and other components like fuel tank walls [2]. Thus, this verification function computes the distance between the two components and checks if the given minimum distance requirement is met. This evaluation provides crucial information on non-compliant fuel lines and with which fuel tank the requirement is not met. The “minimum distance” requirement can of course also be applied to other types of components.

Components inside tanks

This rule can be used when it is necessary that a component is fully contained inside a fuel tank. A special case of this is a boost pump, that usually must not only be contained, but attached to the bottom surface of the tank. In order to check this requirement, this verification function computes the distance between the pump as the bottom surface as well as the misalignment of the pump’s longitudinal axis to the surface’s normal vector.

Figure 5 and Figure 6 illustrate how Codex can leverage the available geometric knowledge to perform requirements verification. While in both cases the fuel pumps are contained inside the fuel tank, only the fuel pump in Figure 5 meets the alignment requirements with the bottom surface of the fuel tank.

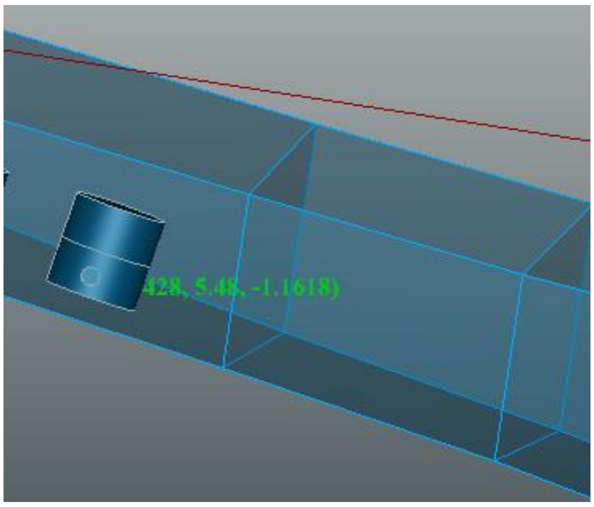


Figure 5 - Boost pump correctly placed in a fuel tank

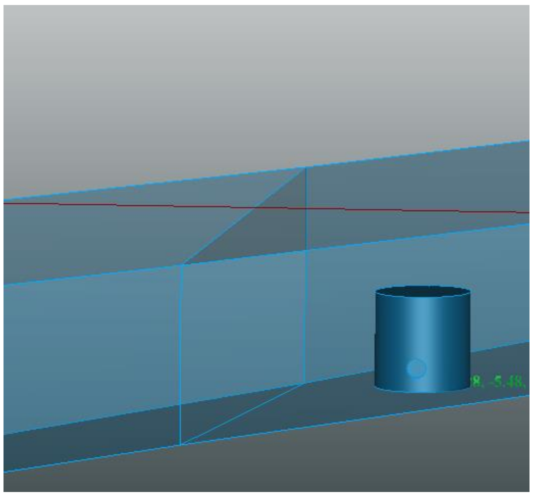


Figure 6 - Boost pump contained in a fuel tank without respecting the (bottom) mounting requirement

Connections between components and supply lines

This rule allows to check if supply lines are properly connected to other components or to each other, e.g. "Front inner pump has to be connected to feed line". In order to verify this requirement, three separate functions are evaluated:

1. The line has to be connected to the proper connection port on the other component, without gaps or intersections.
2. The line has to be correctly aligned with the connections port's normal direction.
3. The size of the connection port has to match that of the supply line.

Figure 7 illustrates an erroneous connection between the supply line (yellow) and the fuel valve. Although the supply line is in fact connected to the supply line, the developed rule is capable of identifying problems with the connection, where the connection point is wrongly placed. Figure 8 shows additional requirements verification for the connection between fuel lines and boost pumps. By exploiting the enhanced geometric information on the components, the verification rule is capable of identifying gaps (or overlaps) and misalignment between the connecting elements.

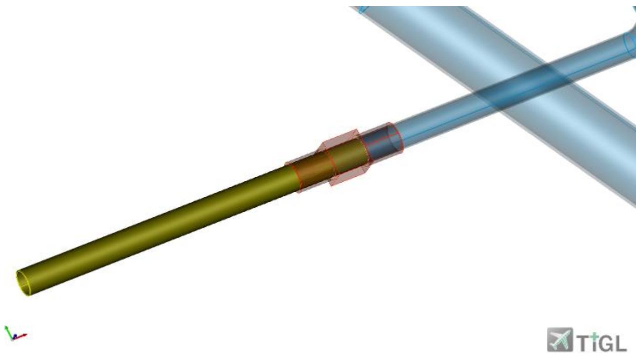


Figure 7 - Connection error between a supply line and a fuel valve

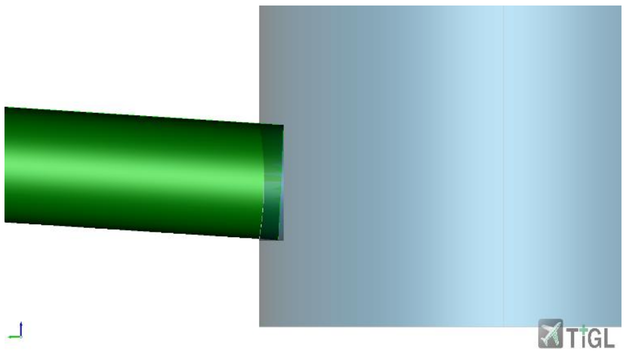


Figure 8 - Geometry mismatch due to misalignment

Relative position between valves and lines

This rule is used to check if a shut-off valve is in an icing-prone condition. Led by Extended range twin operations approval (ETOPS) requirements for cross-feeding purposes, this rule exploits the available geometric knowledge on individual elements and their connections, and provides a preliminary indication of design mistakes that might significantly affect the system's performance. Using information on the relative height between a supply line and a fuel valve, conditions where water accumulation close to the valve could occur are identified. This rule is capable of analyzing fuel lines made by multiple segments at different heights and leverages information on a segment's extreme points and its adjacent segments to identify icing-prone zones for leveled flight.

Figure 9 illustrates the concepts of low and high valve positions in relation to icing-prone conditions. The supply line on the left, shown in red, is configured in a manner that can result in water accumulating near the valve. In contrast, the supply line on the right, depicted in green, is designed to direct water to a lower position away from the valve, thereby preventing accumulation.

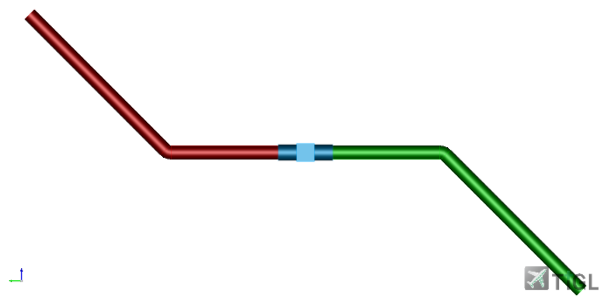


Figure 9 - Fuel valve connected to two supply lines

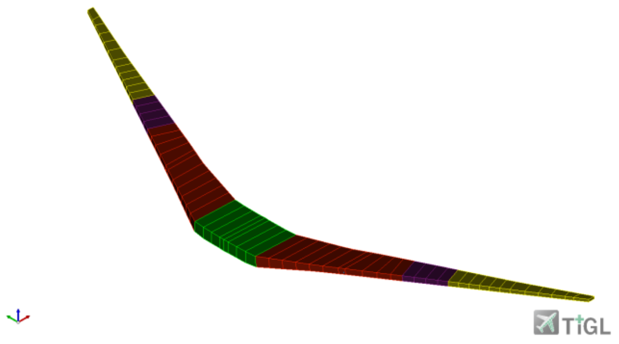


Figure 10 - Wing fuel tanks

Tank volumes

This verification rule is used to check if a required tank volume is available. It is capable of extracting the volume information both for individual fuel tanks and for group of tanks (e.g. left and right feed tanks, all fuel tanks together) and compare it to the specified volume requirements. Figure 10 displays the design space generated for each of the fuel tanks for the example aircraft, based on defined volume requirements. From most outer to most inner, the fuel tank design spaces are as follows: surge tanks (yellow), outer wing tanks (purple), inner wing tanks (red), and center tank (green). The design space is generated, and afterwards validated, using the structural information previously defined and available (i.e., spars and wing ribs positioning).

Available fuel mass

Two verification rules were implemented to determine if a minimum required fuel mass is available. One rule one focuses on the entire fuel system (all fuel storage components) and the other on individual tanks. EASA CS-25 specifications and guidelines are translated into quantifiable code and allow adequate fuel mass estimations based on the available tank volume and considering standard fuel expansion parameters for Jet A and Jet A-1 fuels, and the recommended sump volume.

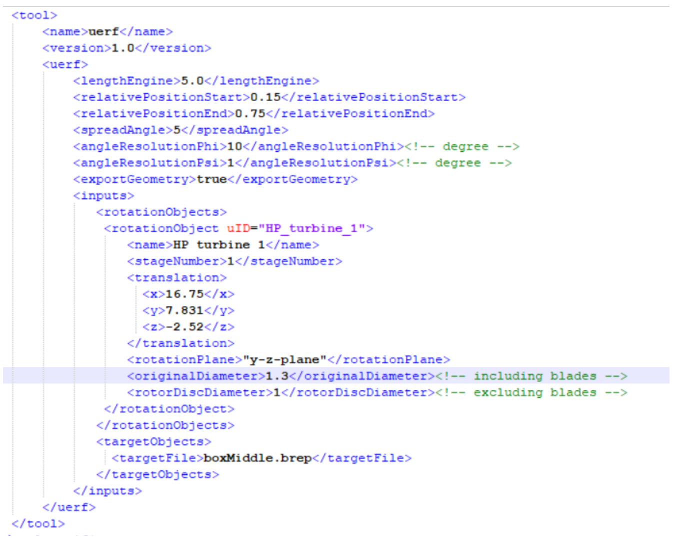
Overall, 430 checks were carried out on the example configuration, which completed in approximately 5 minutes on a standard personal computer. A full report on those can be found in [20].

4. Checking the impact of an uncontained rotor burst

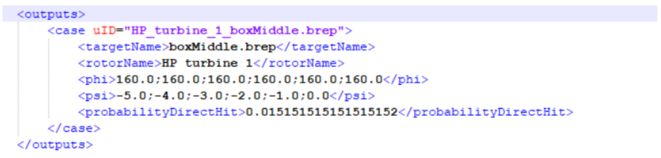
The previous section introduced several design requirements concerning the relative positioning of different fuel system components to each other. As the fuel system is highly integrated with other aircraft subsystems, it is equally important to consider interactions with those other subsystems. An important application case for the automatic evaluation of geometric requirements is the case of an uncontained rotor burst. Following the recommendations of [21], to minimize the risk of engine burst hazards, critical components should be either located outside the debris impact zones or duplicated such that not both of the redundant components are inside the same impact zone.

Based on codex-geometry and on the fuel system model described in the previous chapters, an application for assessing the risk of a rotor burst for critical components was created. The application takes as input a set of critical components and a set of rotors and estimates the probability of each critical component to be hit by a burst of each rotor, considering different spread angles and trajectories.

Figure 11 shows the tool-specific input and output for the tool in the CPACS file. In the input file, first the length and relative position of the engine are specified. The spread angle specifies which angle of spread (fore and aft) to consider. Depending on the type of debris different spread angles are chosen. For intermediate fragment sizes a 5° spread angle is chosen [21]. The two “angleResolution” parameters control the granularity of the collision check. So, in this example, the rotational angle φ will be iterated from 0° to 360° in 10° steps and the spread angle ψ from -5° to +5° in steps of 1°. For each rotation object and each combination of φ and ψ, the trajectory will be computed and the tool will check if the trajectory overlaps with one of the given critical components.
For simplicity, this example shows only one rotation object (typically several stages of a turbine would be considered) and one target object representing a critical component, which can be imported from a BREP file or specified directly in the CPACS file.



```
<tool>
    <name>uerf</name>
    <version>1.0</version>
    <uerf>
        <lengthEngine>5.0</lengthEngine>
        <relativePositionStart>0.15</relativePositionStart>
        <relativePositionEnd>0.75</relativePositionEnd>
        <spreadAngle>5</spreadAngle>
        <angleResolutionPhi>10</angleResolutionPhi><!-- degree -->
        <angleResolutionPsi>1</angleResolutionPsi><!-- degree -->
        <exportGeometry>true</exportGeometry>
        <inputs>
           <rotationObjects>
            <rotationObject uID="HP_turbine_1">
               <name>HP turbine 1</name>
               <stageNumber>1</stageNumber>
               <translation>
                 <x>16.75</x>
                 <y>7.831</y>
                 <z>-2.52</z>
               </translation>
               <rotationPlane>"y-z-plane"</rotationPlane>
               <originalDiameter>1.3</originalDiameter><!-- including blades -->
               <rotorDiscDiameter>1</rotorDiscDiameter><!-- excluding blades -->
            </rotationObject>
           </rotationObjects>
           <targetObjects>
            <targetFile>boxMiddle.brep</targetFile>
           </targetObjects>
        </inputs>
    </uerf>
</tool>
```



```
<outputs>
    <case uID="HP_turbine_1_boxMiddle.brep">
        <targetName>boxMiddle.brep</targetName>
        <rotorName>HP turbine 1</rotorName>
        <phi>160.0;160.0;160.0;160.0;160.0;160.0</phi>
        <psi>-5.0;-4.0;-3.0;-2.0;-1.0;0.0</psi>
        <probabilityDirectHit>0.015151515151515152</probabilityDirectHit>
    </case>
</outputs>
```

Figure 11 - Tool-Specific input and output for the rotor burst tool in the CPACS format

In Figure 12 the trajectories hitting the given critical component are depicted. The output file (see Figure 11) lists all the combinations of φ and ψ that lead to a hit. The probability of a direct hit is estimated by dividing the detected hits by the overall number of sample trajectories that were checked.

The presented tool can easily be expanded to other use cases involving impact trajectories, like a tire burst case.

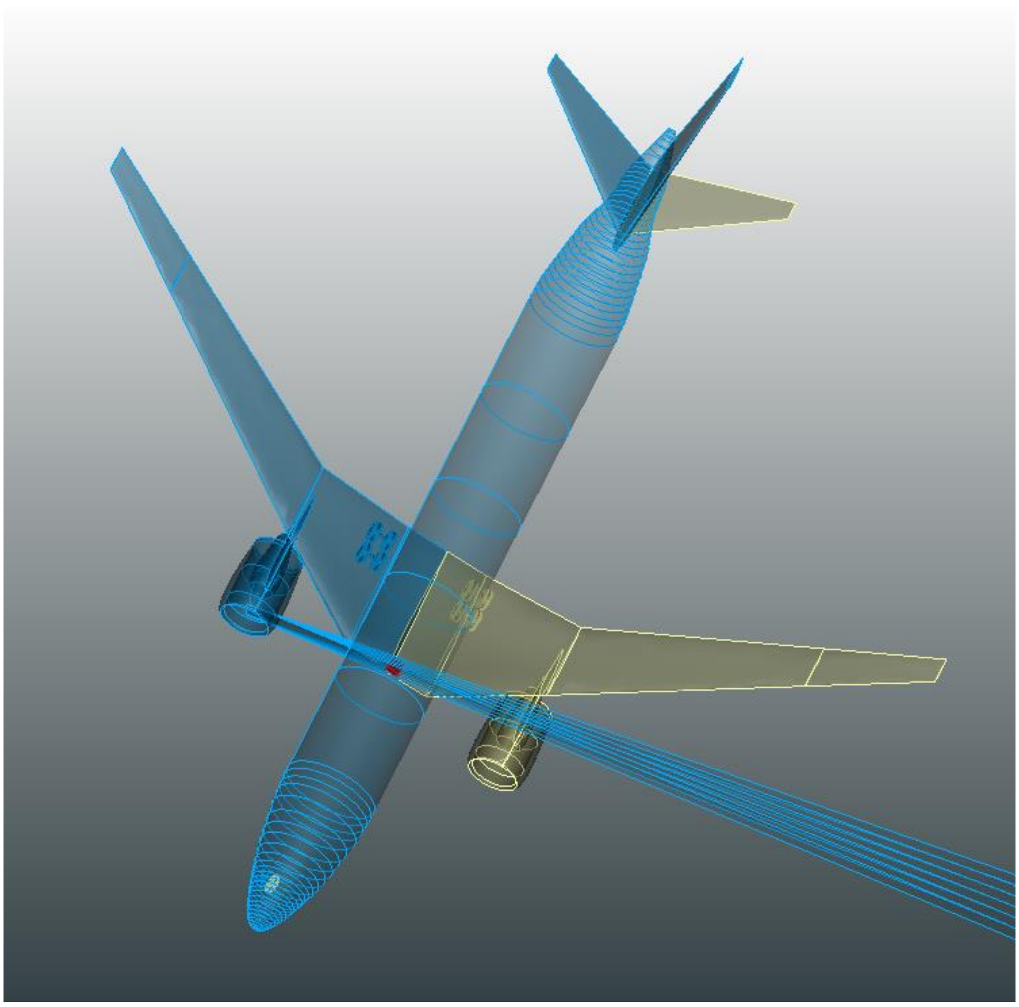


Figure 12 - Result of the rotor burst tool highlighting critical angles for an example component and one turbine

5. Conclusion and Outlook

This paper presented a rule-based approach for the verification of geometric and certification requirements using the semantic web-based KBE platform Codex. Using its geometric capabilities, we created a detailed geometric model of the fuel system and specified geometric requirements derived from certification specifications and guidelines. Using the Codex rule system, the compliance of the geometry with the given requirements was automatically checked.

In conclusion, due to is large volume and high integration with other systems, the geometric design of the fuel system highly impacts the overall performance of the aircraft. By incorporating engineering knowledge as well as design guidelines and requirements into geometric rules and performing automatic consistency checking, the proposed system contributes to avoiding errors in the geometric design already in early design stages. Thus, the application of the Codex platform can help designers save time and effort for these tasks, allowing them to concentrate on the creative part of their design tasks.

Applying these checks is a first step towards an automated design of fuel systems. In our future work we plan to approach the automatic placement of components based on user-defined requirements and target functions. A next step towards this goal could be to start with a single component for which a requirement is violated and suggest an improved position for that component. A further step would be to automatically suggest a placement for a set of components that comply with all given requirements.

Additionally, for the design of fuel systems the routing of the supply lines is a very important and complex problem. Suggesting or improving such routes automatically will also be a topic for future work.

6. Contact Author Email Address

brigitte.boden@dlr.de

7. Copyright Statement

References

[1] European Union Aviation Safety Agency, 2021. *Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes (CS-25) Amendment 27: CS-25*.

[2] SAE International, 2019,"Aircraft Fuel System Design Guidelines AIR7975,"

[3] Allemang D., and Hendler J. A., 2012. *Semantic web for the working ontologist: Effective modeling in RDFS and OWL,* 2nd ed., Elsevier, Morgan Kaufmann, Amsterdam, Boston, Mass., Heidelberg, London, New York, NY, Oxford, Paris, San Diego, Calif., San Francisco, Calif., Singapore, Sydney, Tokyo.

[4] Zamboni J., Zamfir A., and Moerland E., 2020 - 2020,"Semantic Knowledge-Based-Engineering: The Codex Framework," *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, SCITEPRESS - Science and Technology Publications, pp. 242–249.

[5] Jepsen J., Zamfir A., Boden B., Zamboni J., and Moerland E., 2023,"On the Development of a Collaborative Knowledge Platform for Engineering Sciences," *15th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pp. 208–215.

[6] Boden B., Cabac Y., Burschyk T., and Nagel B., 2022,"Rule-based Verification of a Geometric Design using the Codex Framework," *33rd Congress of the International Council of the Aeronautical Sciences, ICAS 2022*.

[7] Krüger W. R., Gerlinger B., Brodersen O., Klimmek T., and Günther Y., 2020,"Das DLR-Projekt KonTeKst: Konzepte und Technologien für emissionsarme Kurzstreckenflugzeuge," *DLRK 2020 - Deutscher Luft- und Raumfahrtkongress*.

[8] TXT Group,"Pacelab APD & SysArc," https://pace.txtgroup.com/products/preliminary-design/pacelab-apd-for-engineers-by-engineers.

[9] ParaPy B.V.,"The ParaPy Platform," https://www.parapy.nl/features/.

[10]Schaefer J., and Rudolph S., 2005,"Satellite design by design grammars," Aerospace Science and Technology, **9**(1), pp. 81–91.

[11]Jens Schmidt, 2017,"Total Engineering Automation: Vision and Realization with Graph-based Design Languages and the Design Cockpit 43 R© software suite,"

[12]Vogel S., and Stephan Rudolph, 2016,"Automated piping with standardized bends in complex systems design," International Conference on Complex Systems Design & Management, pp. 113–124.

[13]Burschyk T., Cabac Y., Silberhorn D., Boden B., and Nagel B., 2023,"Liquid hydrogen storage design trades for a short-range aircraft concept," CEAS Aeronaut J, **14**(4), pp. 879–893.

[14]Zamboni J., Zamfir A., Moerland E., and Nagel B., 2022,"A semantic knowledge based engineering framework for the rapid generation of novel air vehicle configurations," *33rd Congress of the International Council of the Aeronautical Sciences, ICAS 2022*.

[15]"Open CASCADE," https://old.opencascade.com/content/overview.

[16]S-18 Aircraft and Sys Dev and Safety Assessment Committee,"Certification Considerations for Highly-Integrated Or Complex Aircraft Systems,"

[17]S-18 Aircraft and Sys Dev and Safety Assessment Committee,"Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment,"

[18]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.

[19]Siggel M., Kleinert J., Stollenwerk T., and Maierl R., 2019,"TiGL: An Open Source Computational Geometry Library for Parametric Aircraft Design," Math.Comput.Sci., **13**(3), pp. 367–389.

[20]Padilha M. H., 2023,"Enabling fuel system component placement on the basis of geometric and certification requirements within a semantic knowledge-based engineering framework," Politecnico di Torino.

[21]Federal Aviation Administration, 1997. *Design Considerations for Minimizing Hazards caused by Uncontained Turbine Engine and Auxiliary Power Unit Rotor: AC 20-128A*.