PARAMETRIC DATAMODEL FOR COLLABORATIVE PRELIMINARY AIRCRAFT ENGINE DESIGN

S. Reitenbach*, C. Hollmann†, J. Schmeink‡, M. Vieweg§, T. Otten¶, J. Häßy‖, M. Siggel**
German Aerospace Center (DLR), Cologne, Germany, D-51147

An essential aspect of the collaborative multidisciplinary preliminary design process of aircraft engines is the involvement of a large number of experts from different disciplines and the usage of numerous tools and workflows. This is a major challenge, especially in the area of data management, as large amounts of data are generated that have to be exchanged and traced via a multitude of interfaces. The purpose of the paper is to present an approach of a central data model as a vehicle for data management in collaborative aircraft engine preliminary design. The first part describes the general structure and abstract definition of the developed data model utilizing modeling languages. Subsequently, a detailed description of the geometric parameterization concerning important engine components and additional data structures follows. Besides the methodology, the software implementation to support this approach is presented in detail, including data serialization and data identification, as well as automated capturing and storage of provenance data throughout the design process. The functionality of the approach is demonstrated by means of examples derived from the preliminary engine design.

I. Nomenclature

API	=	Application Programming Interface
CAD	=	Computer Aided Design
CFD	=	Computational Fluid Dynamics
CPACS	=	Common Parametric Aircraft Configuration Schema
CR	=	Cruise Condition
DLR	=	German Aerospace Center
EIS	=	Entry Into Service
EOF	=	End of Field Condition
FEM	=	Finite Element Method
FN	=	Net Thrust
GTlab	=	Gas Turbine Laboratory
GUI	=	Graphical User Interface
HPC	=	High Pressure Compressor
HPT	=	High Pressure Turbine
ISA	=	International Standard Atmosphere
LPT	=	Low Pressure Turbine
MDAO	=	Multidisciplinary Design, Analysis and Optimization
MTO	=	Maximum Take-Off Condition
PEGASUS	=	Preliminary Gas Turbine Assessment and Sizing
PERFECT	=	Preliminary design and evaluation of future engine concepts
PLM	=	Product Lifecycle Management
T	=	Temperature

*Research Associate, Institute of Propulsion Technology, Linder Hoehe, 51147 Cologne, Germany, stanislaus.reitenbach@dlr.de
†Research Associate, Institute of Propulsion Technology, Linder Hoehe, 51147 Cologne, Germany, carsten.hollmann@dlr.de
‡Research Associate, Institute of Propulsion Technology, Linder Hoehe, 51147 Cologne, Germany, jens.schmeink@dlr.de
§Research Associate, Institute of Propulsion Technology, Linder Hoehe, 51147 Cologne, Germany, maximilian.vieweg@dlr.de
¶Research Associate, Institute of Propulsion Technology, Linder Hoehe, 51147 Cologne, Germany, tom.otten@dlr.de
‖Research Associate, Institute of Propulsion Technology, Linder Hoehe, 51147 Cologne, Germany, jannik.haessy@dlr.de
**Research Associate, Simulation and Software Technology, Linder Hoehe, 51147 Cologne, Germany, martin.siggel@dlr.de

TOC	=	Top of Climb Condition
UCAV	=	Unmanned combat aerial vehicle
UHBR	=	Ultra-High Bypass Turbofan
UML	=	Unified Modeling Language
VSP	=	Vehicle Sketch Pad

INTRODUCTION

Due to the finite nature of fossil fuels, both the conservation of resources and operational efficiency play a significant role. The constantly growing air traffic and the associated negative effects on the environment are increasing, so that minimizing environmental pollution has become a high priority. In addition, the constantly growing market and competition requires an acceleration of the development of new products and innovations. With continuously more accurate and faster simulation techniques, as well as innovative methods in context of multidisciplinary and collaborative efforts, these challenges are already largely addressed in the design of aircraft engines.

According to Ref. [1] the development of an aero-engine can be divided into three phases, including the preliminary design, the detailed design, and the full production design. Especially the high degree of freedom in the early design phases inevitably implies that many iterations are performed for a given engine design [2]. This multidisciplinary, collaborative, and highly iterative process is handled by a large number of participants, tools and workflows, and most importantly requires an efficient and consistent infrastructure to transfer data and information [3]. The transition to the detailed design or full production design and vice versa is also of great importance. The more consistent a model is, the more time is saved in the transformation and interpretation of the data.

An accurate, parametric, and standardized description of the geometry in the collaborative design process is a key aspect, especially for highly-integrated propulsion systems where consideration of both the inner and outer geometry of the engine is necessary. Interactions between nacelle, wings and airframe, as well as restrictions in diameter and length (drag and required space) and weight must be balanced.

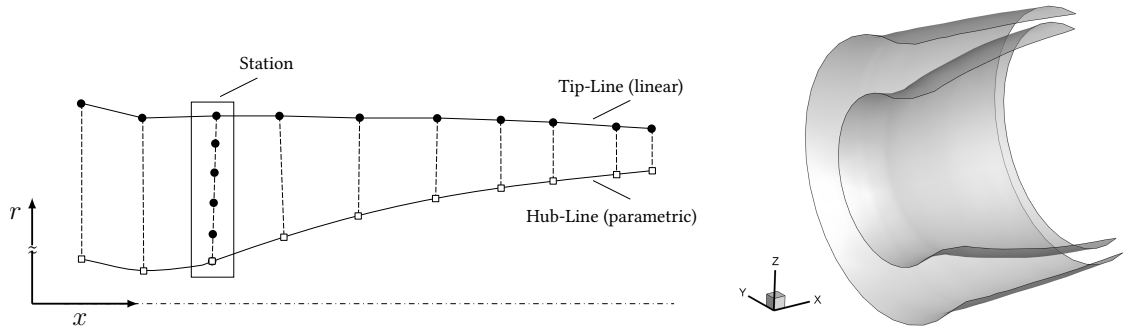
There are several approaches to data modeling and geometric parameterization in collaborative processes in the literature. Debbey et al. [4, 5] presented a methodology for a parameterization of the propulsion system utilizing the tool Vehicle Sketch Pad (VSP) [6]. The parameterizations of 3D geometries are intended to facilitate a more detailed analyses of flow-path design for highly integrated propulsion-vehicle concepts in context of Multidisciplinary Design, Analysis and Optimization (MDAO).

A parametric geometry definition of jet exhausts for future civil aero-engines based on class-shape transformation was presented by Goulos et al. [7]. This integrated approach was coupled with an automatic mesh generation and an aerodynamic analysis. Heidebrecht et al. [8] used class-shape transformations curves to develop a parametric geometry definition for turbofan nacelles in the context of preliminary design.

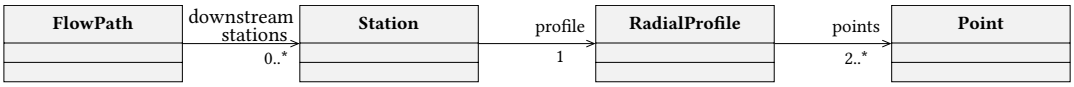
In addition to the field of aero engines, there are also various approaches in multidisciplinary preliminary aircraft design. Kulfan [9] described a parametric geometry airfoil representation method, which can also be applied to other geometries, such as nacelles, ducts and wings.

Böhnke, Nagel and Gollnick [10] present a procedure for the automated conceptual preliminary aircraft design using a distributed design environment. The chosen infrastructure for data processing was realized by a hierarchically structured data model, the Common Parametric Aircraft Configuration Schema (CPACS) [11].

The purpose of this paper is to present an approach of a parametric data model that serves as a common language in collaborative preliminary aircraft engine design of the German Aerospace Center (DLR). The engine system is broken down into its different components, which are parameterized by their individual characteristics. The strict modeling of dependencies between the components enables the creation of individual assemblies, up to the overall engine system. In contrast to previous approaches in the literature, the focus is not only on the modeling itself, but also on the software-technical implementation. An operating system independent software library, including an Application Programming Interface (API), allows for an efficient management of the intricate data flows during design processes and thus helps to handle the extensive amount of transferred data between different disciplines and fidelity levels in a standardized manner. The history of the generated or transformed data can be captured by the library using metadata information and stored via a provenance data management system. This can help to trace and, if necessary, undo any changes made to the data sets in the course of the design process. Finally, the capabilities offered by the central data model approach with respect to the design and representation of the engine geometry are presented.



(a)



(b)

Fig. 1 Flowpath definition

DATA MODEL IMPLEMENTATION

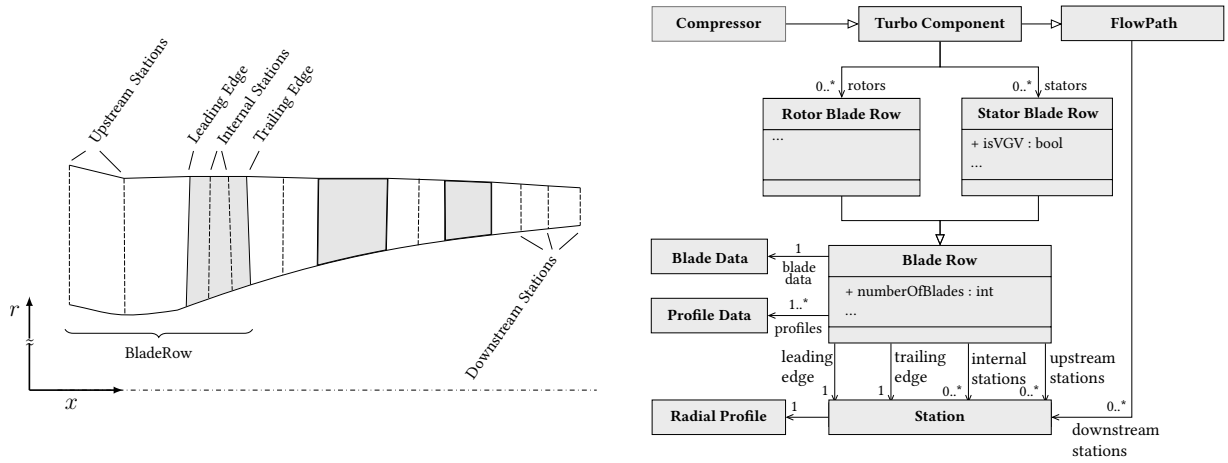
The implementation of the central data model approach can be divided into two major categories: the standardized structure including the specific modeling of each aircraft engine component and the software implementation for integration into the design and simulation processes of the collaborative preliminary aircraft engine design.

In order to define the precise structure of the data model, the unified modeling language (UML) is utilized. UML is a well-established, object-oriented, and standardized modeling language which is ideally suited for specification, description, documentation, and visualization of complex systems. A detailed overview of UML is given by Ref. [12] and the official website provided by Ref. [13]. The most important instrument for the highly abstract representation of engine components and their general relationships in the entire engine system is the class diagram. Thereby each class represents an engine component or subcomponent. Classes can have their own attributes or may inherit attributes from other classes to avoid duplicate definitions. In addition, classes can be connected by associations with other classes to build an assembly. An identifier describes the specific context of the association and the association multiplicity indicates the number of subcomponents that can be available in an assembly. The parameters and attributes of a class are standardized and their functionality clearly specified.

The software implementation of the UML-based data model is realized by a C++ API providing a variety of predefined functions to access the data sets for browsing, reading and writing. The interface has been designed in such a way that any preliminary design tool or process can be coupled in order to extract input data from a central data set and to return the output data set created by the tool to the corresponding structures. In addition, the API uses the CAD kernel OpenCASCADE [14] as its modeling back end. Since the central data model contains all necessary information regarding the engine components, the back end enables the automatic generation of 2-D or 3-D geometries. All generated shapes use watertight surfaces so that they can be transformed into solid geometries. This has the advantage that physical properties such as volume and moment of inertia tensor can be calculated for each geometry. It therefore allows an accurate estimation of the mass of a single component or assemblies in the early design process. Furthermore the interface contains exports functions to common file formats, including IGES, STEP (ISO 10303 [15]), and COLLADA (COLLAborative Design Activity).

COMPONENT MODELING

In the following, the geometric parameterization and data modeling of important components of the aircraft engine system will be presented. In addition to the detailed description of the individual parameters and their relationships, the UML diagrams associated with the individual components are also visualized to provide a better understanding of the applied data modeling approach. In order not to exceed the scope of this paper, the representation of attributes in the



(a) **(b)**

Fig. 2 Turbo component definition

UML diagrams was largely omitted.

Flow Path and Turbo Components

The most important geometric structure for the aerodynamic design and evaluation of engines is probably the guidance of the fluid through an annulus or flow path. The shape of the annulus is defined in the data model by a fixed formation of flow stations, starting with the inlet station. Each station is described by a radial profile with at least two points for hub and tip. Additional points within the radial profile can be specified to define the radial shape between the hub and tip coordinates. The shape of the annulus walls can be defined by connecting all hub and tip points. Without further specification, a straight line from point to point is obtained. This is especially sufficient for a consideration in the course of the conceptual design, since geometrical information between the flow stations are usually not considered at this point. In order to achieve a smooth shape of hub and tip and also to clearly define the shape between the flow stations, it is possible to define additional interpolation parameters which describe the shape by a parametric curve. An interpolation algorithm according to Ref. [16] is used, which generates a B-spline curve $C(u)$ of degree p in the form of the equation (1) based on the data points of the hub or tip.

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i \tag{1}$$

$N_{i,p}(u)$ are the basic functions and P_i are the control points of the B-spline curve. The parameter u is used to determine the corresponding axial and radial coordinates. Furthermore, both the slope and the curvature can be specified in the coordinates of the entry and exit station. Figure 1a shows a flow path with the associated stations. For illustration purposes, the tip contour in this example is defined as a linear connection of all tip points. For the hub contour interpolation parameters were used to describe the shape over a parametric curve. These additional parameters contain the degree of the parametric curve as well as information about the derivatives at start and end points. Thus, the slope as well as the curvature of the curve can be influenced.

The associated UML class diagram for the simple annulus (*FlowPath*) is composed of the individual elements (*Station, RadialProfile, Point*) and the corresponding attributes (Fig. 1b). The hierarchical structure is defined by direct association. Multiplicity allows the abstract relationships of the components to be determined. Accordingly, the annulus can be defined by any number of flow stations (*Stations*). Each station must contain exactly one element of the type *RadialProfile*. This in turn consists of at least two points (*Point*).

The parametric description of the annulus serves as a foundation for the modeling of the turbo components (fan, compressor, turbine). Within the data model, a turbo component is composed by a list of rotor and stator blade rows. A blade row (*BladeRow*) in turn consists of various flow stations which are divided into different categories (Fig. 2a). The most important stations for the blade geometry are the leading (*LeadingEdge*) and trailing edge (*TrailingEdge*).

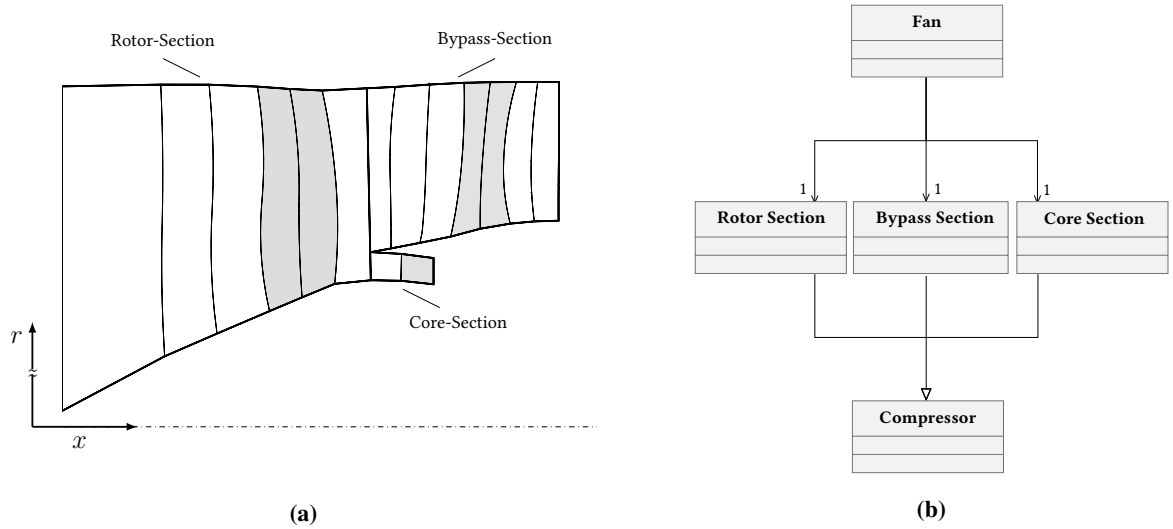


Fig. 3 Fan component definition

These define where the actual geometry of the blade begins and where it ends. For the contouring of the annular space upstream of the blade, further flow stations, the so-called *Upstream Stations*, can be defined. The contouring of the annular space between the leading and trailing edge of the blade row can be defined by specifying *Internal Stations*. The contouring of the blade rows and thus of the flow stations forms the annulus of the turbo component. The extension of the annular space downstream of the last blade row can be defined analogous to the definition of the simple flow path by specifying additional flow stations.

Figure 2b shows the corresponding UML class diagram. The structure of the turbo component (*Turbo Component*) consists of both rotor and stator blade rows. The number of the respective blade rows can be between zero and *n*. In order to avoid double definitions, both structures inherit from a parent class *BladeRow*, but can extend the general definition by additional specific structures. For example, the definition of the stator blade rows contains a flag for variable guide vanes (VGV). Both the class *BladeRow* and the class *Turbo Component* are abstract classes. This means that by definition they cannot be instantiated. An abstract class serves only as a structural element to avoid double definitions. Thus, both the compressor component and the turbine component inherit from the class *Turbo Component* and thus their entire definition.

According to the definition of the central data model, the fan component consists of a rotor, bypass and core section (Fig. 3). The sections themselves are derived from the Compressor component and inherit all relevant structures according to Fig. 2.

Blades

The profile definition as well as the radial distribution of blades depends strongly on the level of detail of the design. While for a mean line calculation mainly simple data e.g. inlet and outlet angles are generally sufficient, a through-flow calculation also requires additional parameters e.g. the variation of the blade height. Whereas 3D-CFD requires the complete description of the blade geometry. For this reason, two abstract class structures have been created in the central data model to enable a multi-fidelity description of the blade geometry.

The abstract class *Profile Definition* is used to store information about airfoils. The abstract class *Blade Definition* is used to store overall parameters of the blade. Each instance of the *Blade Definition* class can contain several instances of the *Profile Definition* class by definition, but at least one. The various parameterizations, which depend on the level of detail, are encapsulated in separate data classes and are derived from the abstract definitions. A blade row can contain different definitions of the blade parameterization depending on the level of detail.

Besides parameterizations for mean line and through-flow profiles, a very powerful variant is the BladeGenerator parameterization, which is developed at DLR [17]. It features a high-degree of freedom in global and local geometry modification as well as the design of curvature distributions to influence shock-boundary layer interactions. Since the parameterization supports automated blade optimization processes for fans, compressors and turbines using

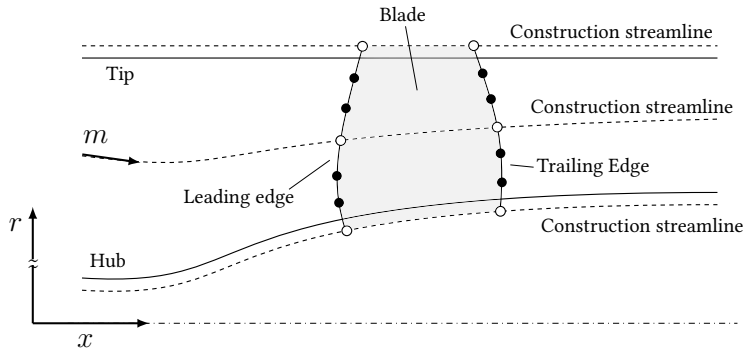
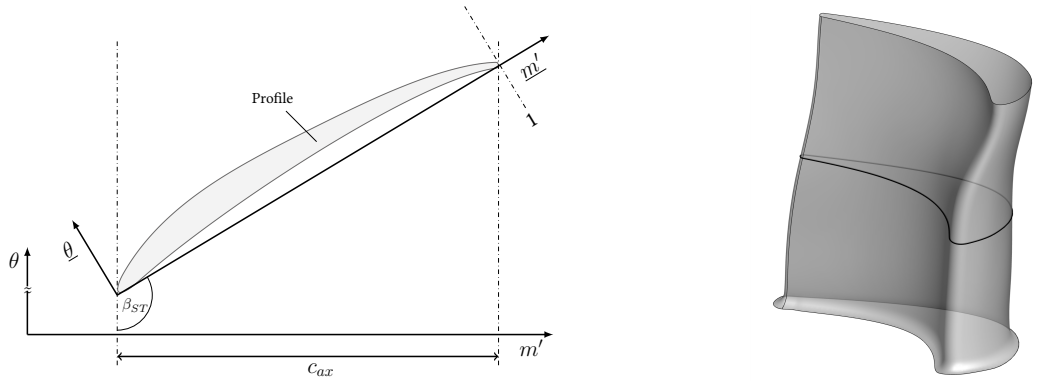


Fig. 4 Blade geometry construction lines



(a) **(b)**

Fig. 5 Blade profile definition and 3D geometry

three-dimensional (3-D) computational fluid dynamics (CFD), the geometry provided in the preliminary design can then be used in tools with higher fidelity without subsequent modifications.

The blade profiles are constructed from multiple B-spline curves in 2-D coordinate system (Fig. 5a). The suction side, the pressure side, the leading edge as well as the trailing edge are defined by individual B-Spline curves and are combined to a uniform B-Spline curve. Subsequently, a coordinate transformation of the blade profiles is performed using the stagger angle β_{ST}.

Since the profiles are designed in 2-D space (m', θ), a construction streamline in (x, r) coordinates is needed for each profile to transform it into 3-D space (Fig. 4). This information is obtained from the parameterization of the *FlowPath* or the *TurboComponent* definition. The radial profiles of the individual *Station* instances related to the *BladeRow* instance are evaluated over the height and are used to construct the required streamlines. The BladeGenerator parameterization supports various additional features for selective modification of the blade geometry, e.g. fillets or shift. An example 3-D CAD model of a turbine blade is shown in Fig. 5b.

Blade Attachments

The blade attachments can be applied to instances of the *Rotor Blade Row* class as optional sub-components. They represent a connection between the blade and the rotor disk. The basic representation of blade attachments in the current state of the engine data model supports three different types apart from the blisk blade to disk connection. Dovetail attachments can be aligned in axial or circumferential direction. Axial dovetails are commonly used in the front stages of high pressure compressors, while circumferential dovetails are typically applied in the rear compressor stages (Fig. 6a). In addition to the two dovetail blade attachments, the fir tree attachment is another common type of blade to disk connection, which is mainly used for turbine blades Fig. 6b).

The description of the attachment, regardless of the type, always consists of three different parts and is based on a set of parameters to describe the geometry without further subordinate data model objects. These descriptions of the

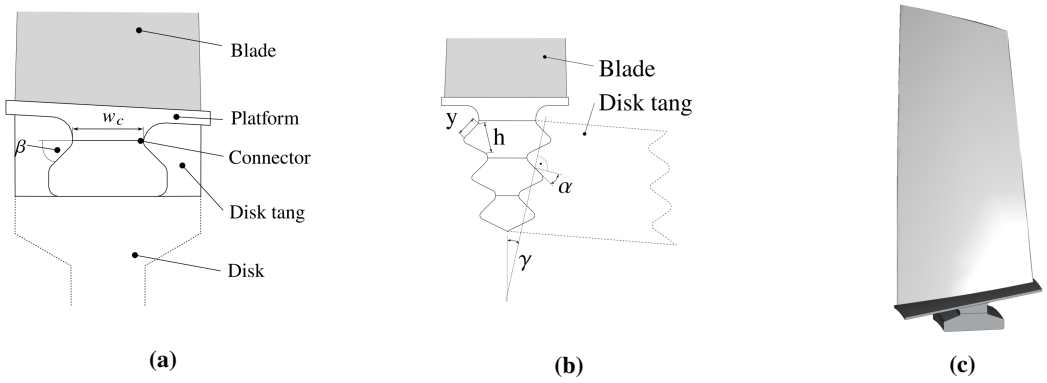


Fig. 6 Geometric description of circumferential orientated dovetail blade attachment and axial orientated fir-tree blade attachment

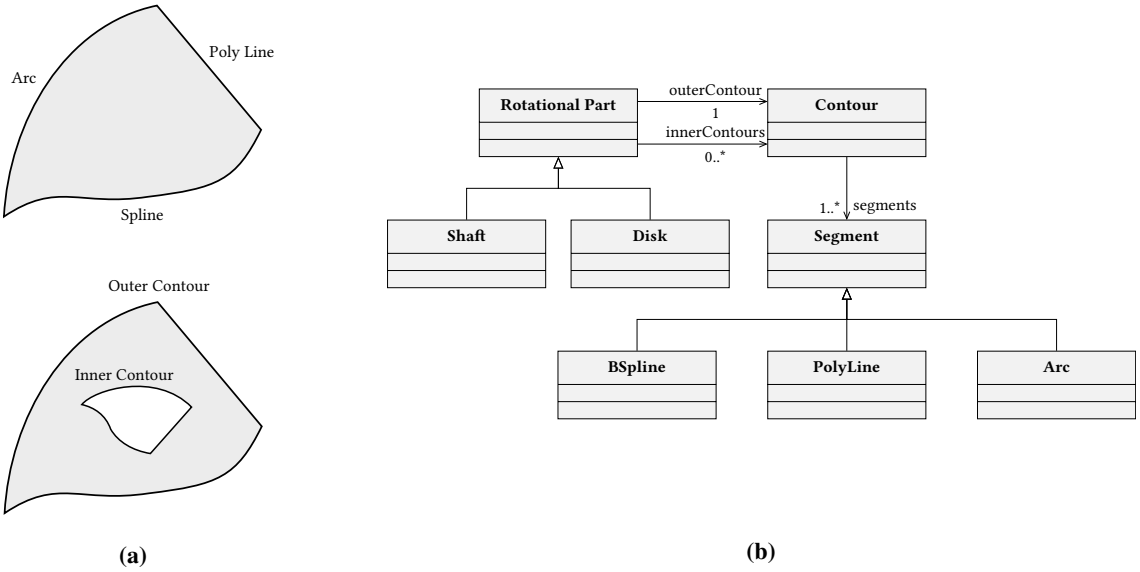


Fig. 7 Definition of rotational parts

attachment are all based on the definitions in Ref. [18] respectively Ref. [19].

All types have the same parameters to describe the platform as the uppermost part of the attachment. Its generic description is fixed by means of one parameter oriented in the radial direction, namely the platform thickness and a width for the axial oriented types (axial dovetail and fir-tree) respectively distances to the blade edges for the circumferential dovetail. The upper shape of the platform is defined by the blade hub shape. The width of the connection to the blade foot is also described for all types using the same parameter, the connector width w_c.

The actual blade foot and the disk tang as counterpart on the side of the disk are parameterized depending on the type. The foot of the dovetail connection attachment is defined by a height in radial direction, a width in axial direction, the angle β (Fig. 6a) and radii for the fillets.

The overall fir-tree blade attachment geometry is defined by the count and the shape of its teeth's. The respective shape is determined by the lengths y and h, as well as by the tooth angle α, the foot angle γ, and the radii of the fillets. A simple example of a three teeth fir-tree connection is shown in Fig. 6b.

The conversion of the 2-D representation derived from the parameterization (Fig. 6a) allows for the transformation to the 3-D domain, as shown in Fig. 6c.

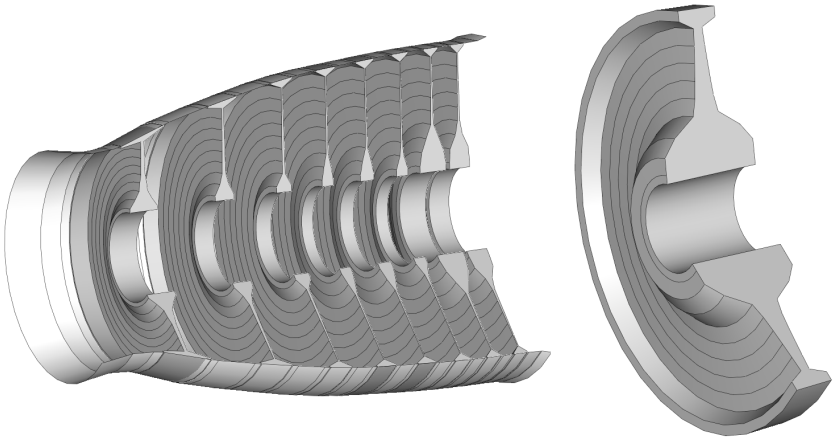
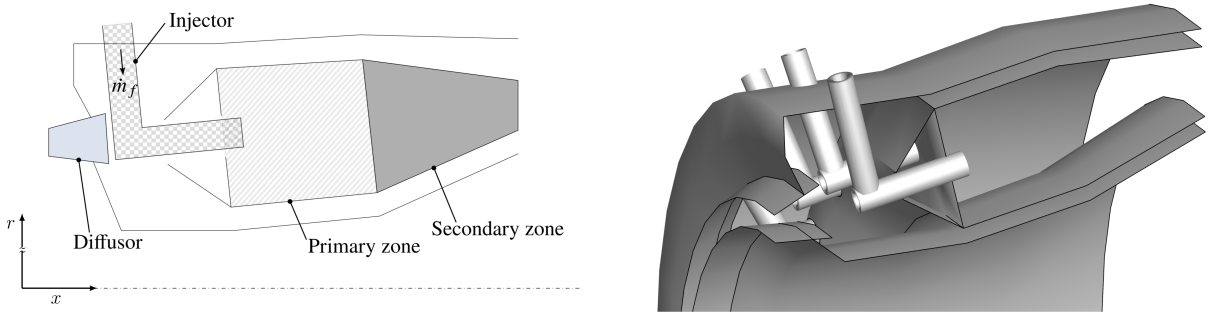


Fig. 8 3-dimensional visualization of disks



(a) **(b)**

Fig. 9 Geometric description of the combustor component

Rotational Parts

Structural-mechanical components that are rotationally-symmetric are parameterized via a standardized data model class called *Rotational Part* (Fig. 7b). By definition, the entire component is composed of multiple instances of the data model class *Contour*. An outer contour to define the outer shape of the component and multiple inner contours to define inner structures like holes (Fig. 7a). A constraint of the contours is that they must be continuous and closed. In addition, the contour may neither intersect itself nor another contour.

The contours themselves are constructed from several segments (*Segment*), which are different types of geometric primitives (e.g. lines, arcs/ellipses, B-splines). It is generally not feasible to further subdivide the segments. The segment *Line* is defined by a simple linear connection of start and end point. The segment *BSpline* is equivalent to the definition of hub and tip lines of the flow path by the equation (1). The arc/ellipse segment is defined by a starting point, an angle of rotation, and the definition of the ellipse (horizonal and vertical diameter, center point).

The connection of the outer contour and the contours of the corresponding holes yield to a surface which by rotation produces the final 3-D geometry. The resulting shape is commonly utilized to generate a 3D finite element mesh. The abstract class *RotationalPart* is the basis for various structural-mechanical components, such as shafts or disks (c.f. Fig. 8).

Combustor

The data model representation of the combustor component is subdivided into several individual parts. It consists of the primary zone, the secondary zone as well as the dilution zone. Each of these zones contains two instances of the data model class *Station*, comprising the inlet as well as the outlet station. Equivalent to *FlowPath* or *TurboComponent*

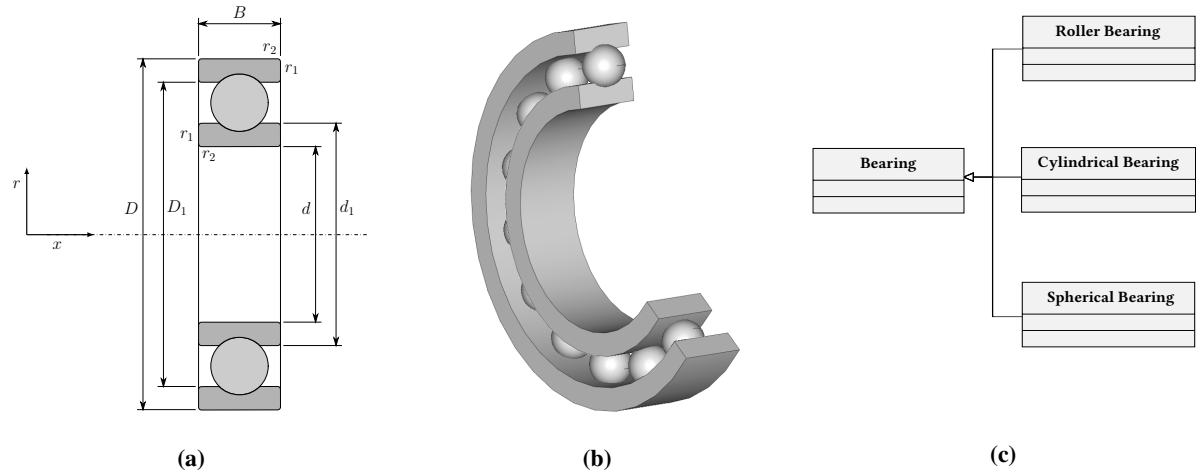


Fig. 10 Geometric description of bearings

class the description of a station contains the definition of radial profile by consecutive points.

The three aforementioned zones are enclosed by the pressure casing, which is divided into an inner and an outer casing, in which a polygon of points can be defined.

Additionally, the diffusor representation belongs to the combustor component, which is basically a description of a duct that can contain struts. In here, next to the struts, an inlet as well as an outlet station can be defined as well as an unlimited number of internal stations.

The injector supplies fuel mass flow into the combustion chamber through pipes. Its representation consists of several geometric parameter that build up its shapes, see Fig. 9a.

The parametric description of the individual zones contains nondimensional parameters that relate height and width of each subcomponent to each other. This eases the scaling process of different designs, whilst the general description within the data model is kept the same. The same parameterization is used to describe reverse flow combustion chambers. An example 3-D CAD model of a combustion chamber component generated from the parametrization is depicted in Fig. 9b.

Bearings

The bearings have the critical function of supporting the engine shafts. The number of bearings necessary is determined by different parameters e.g. length and weight of the shaft. The individual parts of the engine must be held in position within very tight tolerances to ensure efficient and quiet operation while still allowing freedom of motion. To achieve this different type of lubricated bearings are used. Depending on the type, the bearings must take thrust loads, radial loads, or a combination of both.

At the current state, three bearing types are supported by the data model structure, all inheriting from the abstract class *Bearing* (Fig. 10c). Roller bearings are used in applications where radial loads are high. The dimension of the bearing results from the specification of a width B, the outer diameter of the outer ring D, and the inner diameter of the inner ring d (see Fig. 10a). Using equations (2) and (3), the inner diameter of the outer ring D_1 and the outer diameter of the inner ring d_1 can be calculated. The diameter of the rolling element is obtained from equations (4) and (5).

$$D_1 = D - 0.3(D - d) \tag{2}$$

$$d_1 = d + 0.3(D - d) \tag{3}$$

$$E = D_1 + 0.25(D - D_1) \tag{4}$$

$$F = d_1 - 0.25(d_1 - d) \tag{5}$$

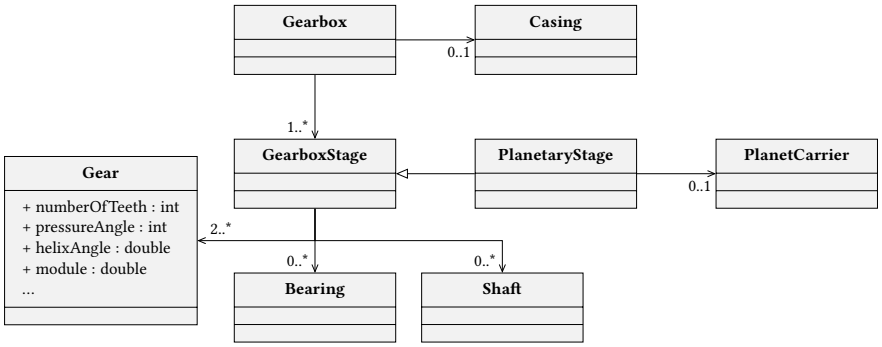


Fig. 11 Gearbox definition

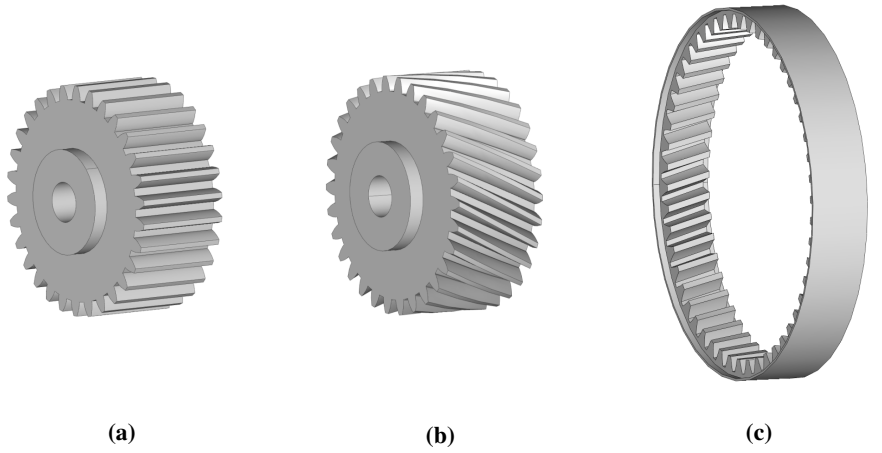


Fig. 12 XML SERIALIZATION, HASHING AND DIFF OF TURBO COMPONENT DATA RECORD

Optionally, the corner radii r_1 and r_2 can be specified. A rotation around the center results in the 3-D geometry of the bearing (Fig. 10b). Unless otherwise specified, the number of rolling elements is considered to be fifty percent of the geometrically possible.

Gearbox

Engine power gearboxes are modeled as an assembly of different parts. Many very different gearbox concepts exist, therefore it is important, that the data model enables an application of these parts in different gearbox concepts.

In the presented data model, a gearbox consists of one or more gearbox stages and an optional gearbox casing. Each stage is modeled independently and can consist of multiple gears, bearings, shafts and - in case of a planetary stage - a planet carrier (Fig. 11).

Gears can be described with inner or outer toothing (required for planetary gears). An involute gearing is assumed, its parametrization is taken from DIN3990 [20] and allows for straight, helical or double helical teeth (Fig. 12). The gear body is described separately by a set of points.

Besides gears, a gearbox system incorporates various bearings. The form of these bearings depend on the gearbox type and a gearbox stage may incorporate various bearings.

For multistage gearboxes, shafts connect the single gear stages. For planetary gear stages, a planet carrier is introduced that connects all plant gears. The planet carrier shape differs significantly between in-service gearboxes, therefore a variety of different carrier designs are applied.

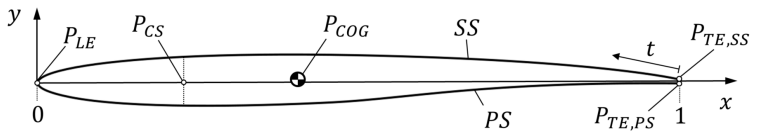


Fig. 13 Shape of a propeller profile defined by a B-spline

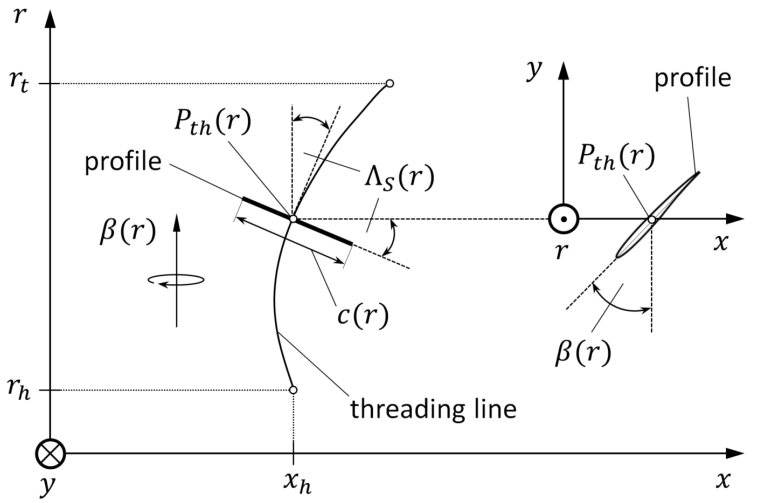


Fig. 14 Definition of the geometrical blade parameters and visualization of the threading line

Propeller

The data model representation of a propeller consists of one or more propeller blade rows enabling single and counter rotation configurations. The position along the blade axis is specified by the nondimensional radius r' (Eq. 6) which is the ratio of the radius r at the current position to the blade tip radius r_t.

$$r' = \frac{r}{r_t} \tag{6}$$

The overall parameters that describe a blade row include the diameter D, the number of blades B, the nondimensional radius at the hub r'_h, the axial position x_h of the threading line at the hub and the direction of rotation which is either clockwise (CW) or counter-clockwise (CCW). Detailed information on the blade geometry is provided by a list of design profiles and by curves that contain geometrical parameters along the blade axis.

The dimensionless 2D-shape of each design profile is parameterized by a B-spline (Fig. 13) whereby the position on the spline is specified by the parameter t. The spline begins at the point $P_{TE,SS} = (1, y > 0)$ at the trailing edge (TE) on the suction side (SS) with $t = 0$ and continues on the SS until the leading edge (LE) point $P_{LE} = (0, 0)$ is reached. From there it returns on the pressure side (PS) back to the TE and ends at $P_{TE,PS} = (1, y < 0)$ with $t = 1$. Furthermore, the center of gravity P_{COG} and the chord share point P_{CS}, which is located at a certain percentage of the profile chord length, are shown.

The position of each design profile along the blade is specified by the radial coordinate $0 \leq r'' \leq 1$ (Eq. 7). At least one design profile is necessary. The propeller data model also defines by which method the profile shape is interpolated at radial positions in between the design profiles.

$$r'' = \frac{r - r_h}{r_t - r_h} = \frac{r' - r'_h}{1 - r'_h} \tag{7}$$

The size, location and orientation of the profile at a specific radial blade position is given by the chord length c, the blade twist angle β and the sweep angle Λ_S (Fig. 14).

In the data model these parameters are stored along the blade as B-spline curves using the radial coordinate r'' as exemplarily shown in Fig. 15. Therefore the chord length is related to the propeller diameter $\frac{c}{D}$.

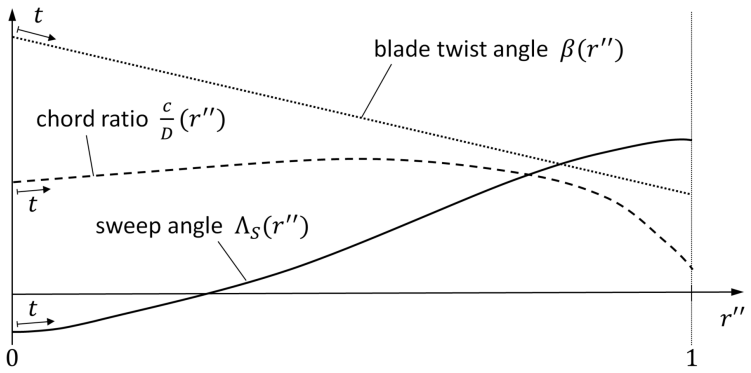


Fig. 15 Qualitative distribution of the geometrical blade parameters along the blade axis parameterized by B-splines

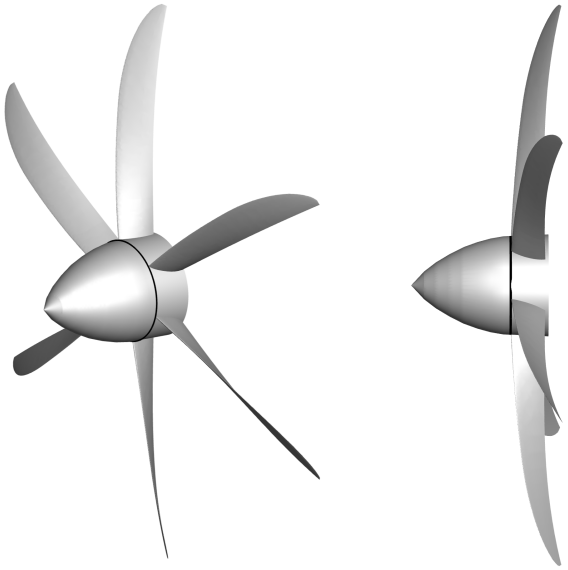


Fig. 16 Exemplary visualization of a six-bladed single rotation propeller defined by the data model including a spinner

The threading points P_{th}, which form the threading line as depicted in Fig. 14, can be calculated by the sweep angle curve and the overall blade parameters. The data model provides threading of the profiles by the center of gravity P_{COG} or by a certain percentage of chord P_{CS}.

Overall the propeller blade geometry is uniquely defined by the data model. At each radial position a dimensionless profile shape is available whereby each profile point P_P can be transformed to a 3D-point of the propeller blade P_B by equation 8 using the rotation matrices Θ_i and P_j, which is either P_{COG} or P_{CS} of the profile depending on the threading mode.

$$\vec{P_B} = \Theta_{\Lambda_S}\ \Theta_\beta\ c \left[\begin{pmatrix} P_{P,x} \\ P_{P,y} \\ 0 \end{pmatrix} - \begin{pmatrix} P_{j,x} \\ P_{j,y} \\ 0 \end{pmatrix} \right] + \begin{pmatrix} P_{th,x} \\ 0 \\ r \end{pmatrix} \tag{8}$$

The resulting geometry of a six-bladed single rotation propeller is shown in Fig. 16 as an example. Besides the sweep also lean can be considered, which is not described here due to limited extent.

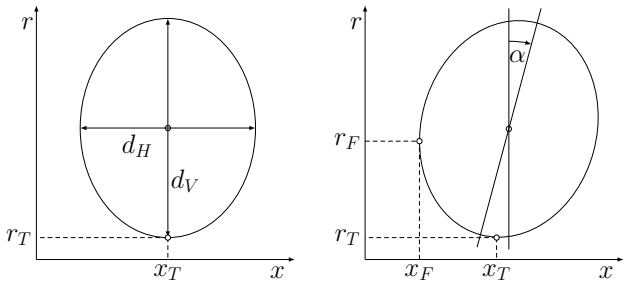
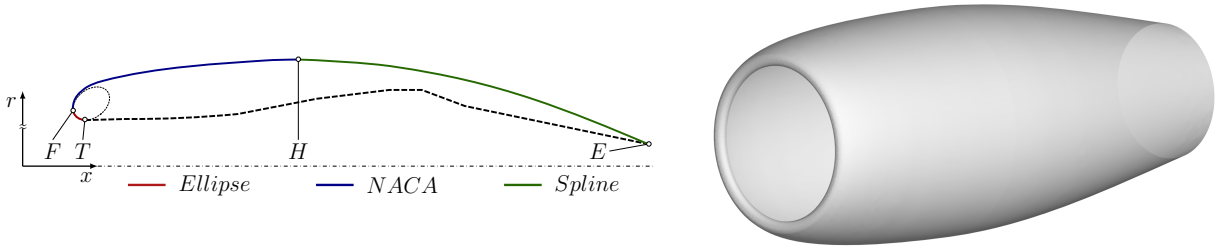


Fig. 17 Definition of the ellipse geometry and positioning in relation to the throat point T



(a) **(b)**

Fig. 18 Definition of rotational parts

Nacelle

A nacelle is a casing separate from the aircraft fuselage, which substantially surrounds the engine. The description of the nacelle is of great importance, especially to consider the interactions between wings and airframe.

Within the data model, the outer contour of the nacelle is parameterized by different points, which divide the geometrical path into a total of three segments. The exact shape of these segments is individually defined and will be explained in the following.

The first point to be defined is the throat point $T(x_{TP}, r_{TP})$ which marks the beginning of the outer contour and is commonly given by the inlet geometry. The first segment describing the nacelle's intake lip starts at this point and is defined by means of an ellipse which is given by the horizontal diameter d_H, the vertical diameter d_V, an angle of rotation α and the position of the throat point T (Fig. 17). The throat point is interpreted as the point with the lowest radial coordinate r. For a given angle of rotation, the point with the lowest radial coordinate of the ellipse is therefore automatically moved into the throat point T. The geometrical segment defined by the ellipse is limited by the foremost ellipse point with the lowest x- coordinate, which is called $F(x_F, r_F)$ in the following.

The other points to be defined are the highest point of the outer nacelle shape $H(x_H, r_H)$ as well as the endpoint $E(x_E, r_E)$ where inner and outer nacelle shapes reunite. The geometrical shape of the nacelle between points F and H is defined by a axis symmetrical NACA airfoil geometry from the NACA Four-Digit Series [21]. With the maximum airfoil thickness T (Eq. 9), the thickness distribution $t(x)$ for the considered segment can be calculated by means of Eq. 10 using the coefficients given in Tab. 1.

$$T = 2(r_H - r_F) \tag{9}$$

$$t(x) = \frac{T}{0.2}\left(a_0 x^{0.5} + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4\right) \tag{10}$$

The last segment extends between points H and E and is approximated by a B-spline. All points and segments as well as a resulting nacelle shape are exemplarily shown in Fig. 18.

Table 1 Coefficients for NACA Four-Digit Series geometry calculation

Coefficient	Value
a_0	0.2969
a_1	-0.126
a_2	-0.3516
a_3	0.2843
a_4	-0.1015

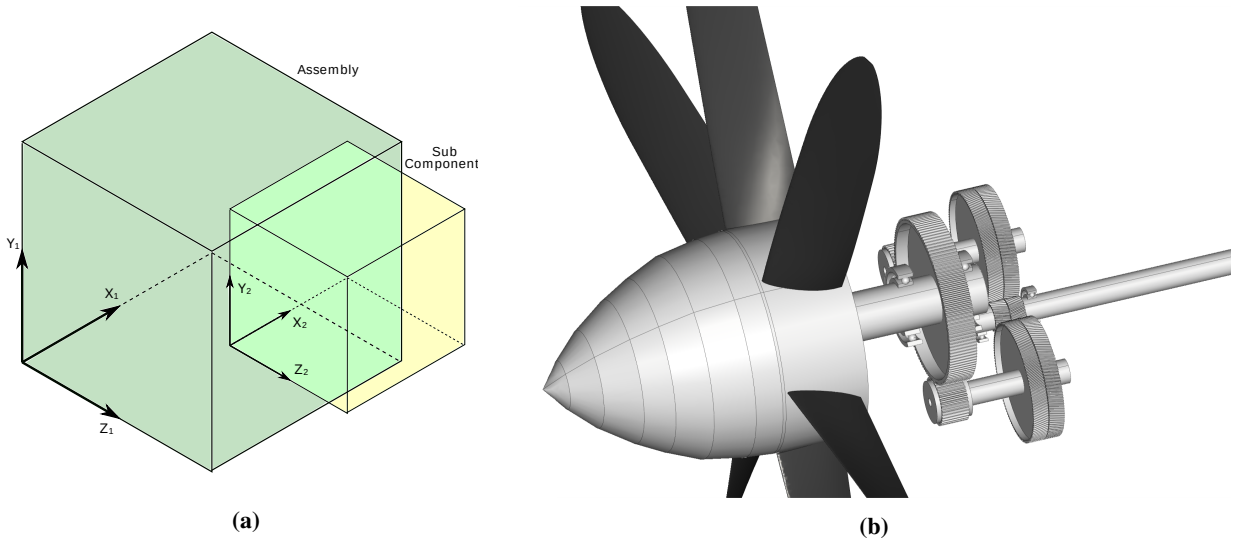


Fig. 19 Component assembly and relative coordinates

Component assemblies

The data model class *Assembly* allows to combine different components into assemblies. The advantage of an assembly is not only an efficient grouping of the individual components, but also its effect on the specific positions in the coordinate system. Each instance of the data model components stores its individual position and thus defines a local coordinate system, wherein all geometrical parameters of the component are defined. For instance, a manual modification of the individual position has a direct impact on all geometric parameters of the component and thus enables absolute control over how the component appear in 3-D space.

When components are added to an assembly as sub-components, they also adopt its coordinate system. All individual component coordinates are then relative to the coordinates of the assembly (see Fig. 19a). Changing the position of the assembly results in a change in the position of the individual sub-components in the context of the global 3-D space. An example of the usage of an assembly component is a gearbox shown in Fig. 19b consisting of multiple gears, shafts, and bearings in combination with a propeller component.

Result Container

The *Result Container* class represents an abstract description of a data element that can hold any kind of specific data (e.g. simulation results). The only function it provides is that labels can be registered as sub elements that serve as an identifier for data. The *Label* class describes under which conditions the result data was created (e.g. operating point identification). Under the label object any number of actual result elements can be stored. These are so called data zone objects. A *Data Zone* is an abstract class, which summarizes functionalities and parameters that all kind of data zones will hold, which is basically a list of parameters and a list of units. From this class two more specific data zone classes are derived. Zero-dimensional data zones that can store one value for each parameter and general data zones that can hold n-dimensional data.

The zero-dimensional data zone contains a vector of value entries and provides functionalities to set this data completely new, thus overwriting existing data, or to append data to the existing list. Tools that provide this kind of data are for instance a performance synthesis code that stores single parameters such as engine's thrust or fuel flow.

The *n*-dimensional data zone contains a table object as a sub element in which as *n* axes can be defined as dimensions. Several usability functions are available that ease the access to and from this type of result data. A 1-dimensional stress calculation routine of a circumferentially symmetrical disk for instance provides for each radially discretized element the whole set of parameter values and can therefore be stored in this type of result data. But also 3-D CFD calculation methods provide data that can be store in there. Internally a table containing three axes representing the geometric space will then be created. Also, time-dependent data can be stored in here.

SOFTWARE IMPLEMENTATION

In order to make the data model structures efficiently accessible to all participants of the engine design process, a software library was implemented which has an interface, for the connection of design or simulation tools and a standardized access to the data. The API is designed in such a way that any preliminary design tool or procedure can be coupled to it in order to extract input data from a central data set and to feed the output data set created by the tool back into the corresponding structures (Fig. 20). The individual data structures within the software library are always based on the UML representation introduced in the previous chapter. The software library is basically based on the programming language C++, but also offers the possibility of access via commands of the programming language Python.

Within the C++ library, each UML component of the central data model is represented by a C++ class. The individual parameters are implemented as member variables with associated read and write methods. For the navigation through the hierarchical structure of the data model, various functions are available. For example, the relationships of the components, which are represented as associations in the UML notation, are implemented as functions within the C++ class. Furthermore, functions are implemented to search the data model directly for specific objects using a recursive algorithm.

During the entire preliminary design phase, the engine model goes through a number of different design phases and undergoes a considerable number of modifications. On the one hand, it is important to be able to track the changes at all times in order to understand what exactly has been changed, when it was changed and possibly why the change was made. On the other hand, it is often necessary to go back several design steps to undo design errors. Furthermore, it is often necessary to be able to work through several design paths simultaneously. The latter occurs if no concrete design decision has been made and several design paths are available. This problem also exists in the field of software development. Version control systems (VCS) have been established as a solution. These systems can be sorted into local, centralized and distributed version control. Especially in collaborative design processes where many data sets are stored, transferred and distributed, methods for data serialization and identification must be provided.

Within the C++ library, each component class is derived from an abstract data class and thus inherits different,

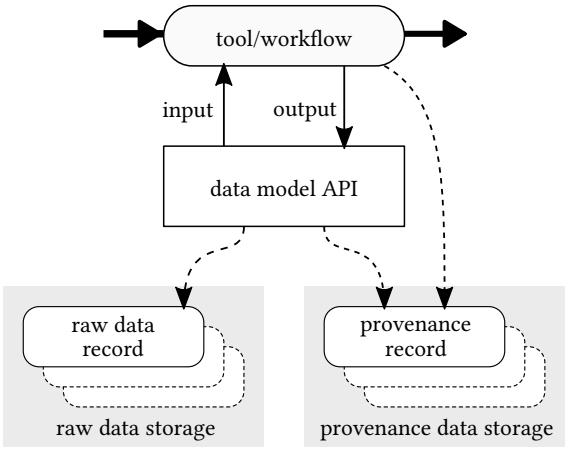


Fig. 20 Data model API

Fig. 21 XML SERIALIZATION, HASHING AND DIFF OF TURBO COMPONENT DATA RECORD

uniform functionalities for data processing. For storing, transferring and distributing data, a process must be provided that enables serialization of the data. In this process the data structure and the internal state of an object are translated into a specific format. For reasons of human readability, the objects of the central data model presented in this paper are serialized into the extensible markup language (XML) format. XML is used as the standard language for data representation in a wide range of applications. However, it is generally possible to serialize into alternative formats (e.g. binary format). An overview and comparison of different serialization formats is given in Ref. [22]. Figure 21a shows a simplified example of the serialization output of a compressor data model component. Each parameter of the component is stored with its current value and thus represents the current state of the component. In addition, if available, the entire hierarchical structure including all child objects of the component is serialized. The serialization process is completely reversible. With an existing serialized data set, an object can be regenerated via the API of the central data model and set to the corresponding state stored in the serialization. The created object can thus be used directly in the subsequent design process.

The unique identification of component within a given dataset is achieved by the automatic assignment of an individual character string at the moment of creation, which remains unchanged during the entire lifetime of the data model including all transformations of the data. This character string is realized by a UUID, which is a standardized method for the unique identification of entities in a distributed computing environment [23].

The current state of a component is identified using a cryptographic hash algorithm. The hash is generated from a serialized data set using SHA-2 standards. The SHA-2 standard is a widely used cryptographic hash function that generates a cryptographic hash from binary or text data [24]. A component of the data model will always have an identical hash as long as its properties and structure remain unchanged. Even the slightest change to the parameters and thus the serialization of the component will result in a modified hash value. Fig. 21a shows the XML serialization of a compressor component including the generated hash. In Fig. 21b a property of the component has been changed, which automatically results in a change of the generated hash value.

To identify the exact change of two data sets, the API provides a diff algorithm. Based on two states of a particular component (same UUID), the diff algorithm recursively identifies changes within the data tree and locates these changes at the lowest possible component level. Typical changes are e.g. added or removed child components and changed property values in the component itself or in its child components. Fig. 21c shows the output of the diff algorithm for the two records shown in Fig. 21a and Fig. 21b. It can be seen that a change of a property has been detected. The diff contains both the new value of the property and the original value before the change.

The ability to uniformly serialize each record of the data model with the unique identification of the current state and to compare it with other records allows to operate various version control systems. Furthermore, a data provenance management system was implemented within the data model API. The acquisition of provenance data is handled via the

data model API utilizing an interface to a data provenance storage (Fig. 20). A tool or workflow receives input data via predefined interfaces and feeds back the generated output after successful execution. All data model components accessed in a process and all related modifications to the data are automatically monitored as metadata entries. The provenance management system provides various predefined query routines for provenance analysis. For example, for a given UUID of a data model component, the entire change history created during the design process can be retrieved.

A detailed description of the implementation of the provenance management system in the existing data model API and its application using a concrete example regarding data inconsistencies is published in Ref. [25].

APPLICATION IN COLLABORATIVE DESIGN PROJECTS

In this chapter the presented approach of the central data model is demonstrated by means of practical examples derived from preliminary engine design projects. Most of the definitions and extensions of the data model and the associated software technical infrastructure are driven by the requirements from these projects, with a focus on the application of multidisciplinary, collaborative design methods to perform complex product design iterations. The main objective is not the aspect of the preliminary design methods (e.g. simulation software or design philosophy) but the demonstration of the possibilities offered by the central data model approach with regard to the design and representation of engine geometry. In addition, the advantages of the approach in terms of the collaborative aspect of preliminary aircraft engine design are highlighted.

Generic Two-Shaft Mixed Flow Turbofan Engine

The first activities on the geometric engine data model go back to previous research activities for DLR's Advanced Technology Research Aircraft ATRA, an Airbus A320 equipped with two IAE-V2527 engines. The engines nacelle and major fan components including disk, casing, outlet guide vanes and booster inlet guide vanes were optically scanned and transferred to surfaced based CAD models in a reverse engineering approach to obtain the geometry in a most accurate manner [26]. Although the initiation was made to enable dedicated CFD studies of the propulsor itself, the digitalized geometries served as input for the DLR preliminary design project PEGASUS. Target of the project was to establish a multidisciplinary predesign process for the assessment of novel aircraft engines to providing a feasible approach toward virtual engine design. The foundation of this collaborative approach should be the development of a central engine data model, whereas the IAE-V2500-A5 was chosen to apply and validate the preliminary design process. For the entire engine and all its components, a cycle model was established in order to derive realistic operating conditions for all components and was validated against experimental data taken by the Electronic Engine Control (EEC) at the aircrafts last shop visit at Lufthansa Technik [27].

Based on the thermodynamic parameters, initial annulus geometries and structural mechanical components of the engine were generated to support the creation of the performance model. Data taken from cross section drawings of the IAE-V2500-A5 engine and the previously mentioned digitalized geometries were used to match dimensioning parameters and to develop initial data model structures. The data model classes derived from this approach were further elaborated during the preliminary design process so that the most important components of the aircraft engine could be represented. The entire architecture of the developed collaborative process and the applied software tools are summarized in Refs. [3, 28]. The final 3-D geometry of the turbofan engine, which was generated entirely from the information of the central data model, is shown in Fig. 22a.

Counter-Rotating Open Rotor

Based on the experiences gained from the preliminary design and validation of the IAE-V2500-A5 engine model, the first application of the design process to future engine technologies in PEGASUS was the counter rotating open rotor engine concept. For the first time, the central data model approach was applied for data exchange. The same top-level aircraft and mission design requirements were defined, to be able to make a comparison between the turbofan engine and the CROR. The propeller blades were designed by an automated optimization process using 3-D CFD, so only the core engine was designed in the preliminary design process. Nevertheless, both the high-fidelity and the low-fidelity geometry models are integrated into the overall engine model based on the central data model. The final design of the compression system consists of a 5-stage low-pressure compressor (LPC) driven by a single stage intermediate pressure turbine (IPT) and a 7-stage high-pressure compressor (HPC) driven by a single stage high-pressure turbine (HPT). Figure 22b shows the final 3-D CAD model of the counter rotating open rotor engine. This engine concept shows the ability of the presented data model approach to describe engines with propellers. The same data classes can

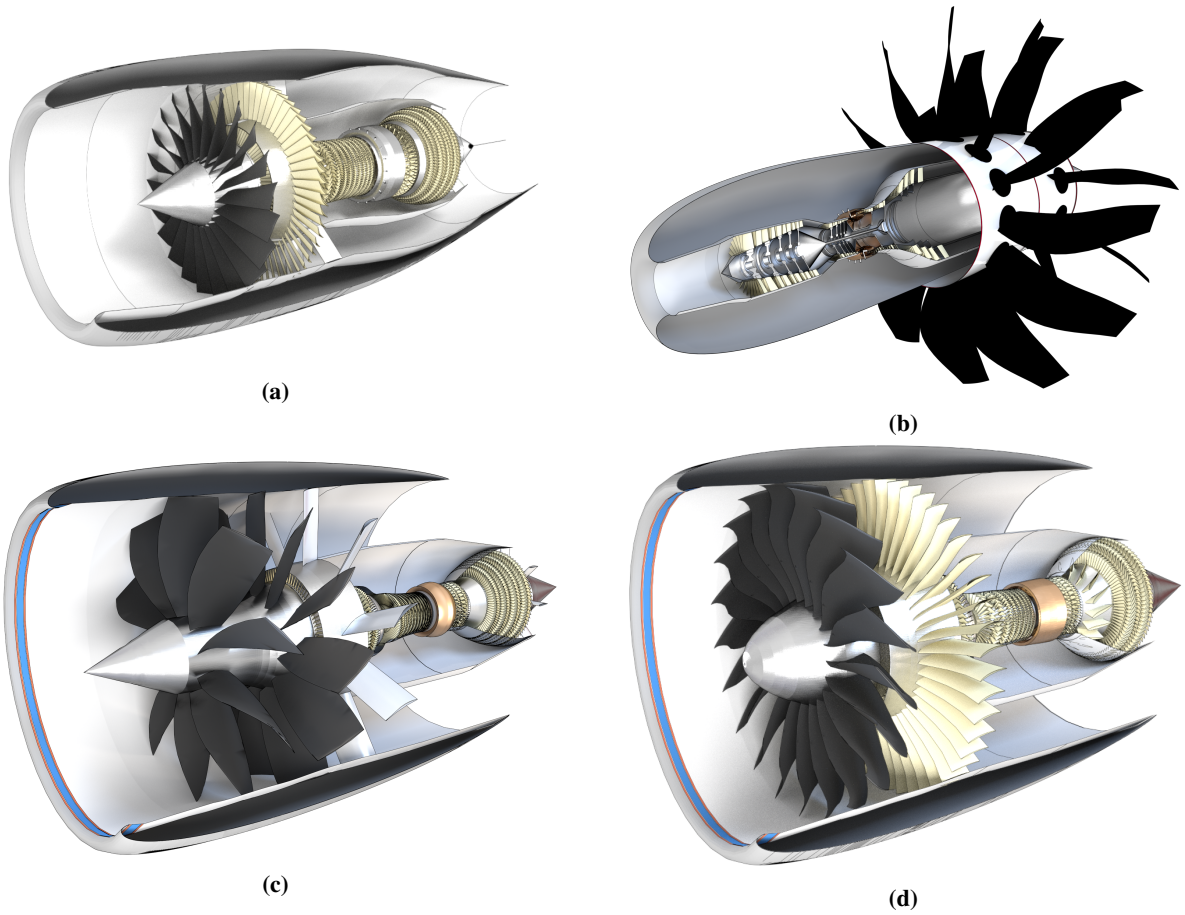


Fig. 22 Aircraft engine configurations investigated in the projects PEGASUS and PERFECT

be used to realize other types of propeller engines.

Ultrahigh Bypass Turbofan

The third engine design of the PEGASUS project was an ultrahigh bypass turbofan concept to power a long-range aircraft configuration oriented on the Boeing 787. The corresponding engine was designed as a counter-rotating turbo fan configuration. Similar to the counter rotating open rotor concept, the main focus was on the preliminary design of the core engine. The propulsor component was designed by a CFD optimization method, which was carried out in a parallel project [29, 30]. Within the overall engine model, the propulsor is driven by a planetary differential gear, which in turn is driven by a four-stage LPT. The gearbox component was only represented in the engine performance model, a geometric representation was not available. The LPC, which is also driven by the low-pressure shaft system, has three stages. On the high-pressure shaft, there is a ten-stage HPC which in turn is driven by a two-stage cooled HPC. The combustion chamber is designed as a lean combustion chamber. The final 3-D CAD model of the ultrahigh bypass turbofan engine is shown in Fig. 22c. This example demonstrates the flexibility of the central data model to integrate innovative components into existing engine design processes.

Generic Geared Turbofan

Based on the experience from the PEGASUS project, the successor project PERFECT (Preliminary design and evaluation of future engine concepts) followed. The target of the project was the enhancement of the existing approaches of collaborative preliminary engine design including the associated central engine data model. The latter was mainly extended by radial compressor, reversed combustion chamber, gearbox, and bearing component data class structures. The aspect of simulation was enhanced by the integration of transient performance calculation and acoustic evaluation.

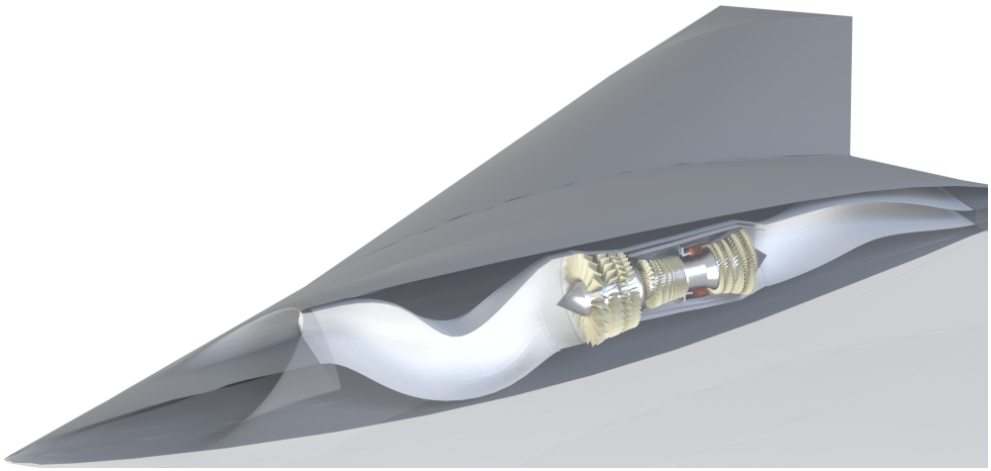


Fig. 23 Military application

In contrast to the PEGASUS project, strict coordination with the preliminary aircraft design team was carried out. All relevant aspects of this enhanced preliminary design process are presented in Ref. [31].

A representative engine model was the generic geared turbofan, based on the top-level aircraft requirements of the reference mission of the A320neo aircraft. This twin-shaft turbofan configuration comprises a gearbox component whereby the low-pressure shaft and the fan run at different speeds. The three-stage LPC is driven by a three-stage LPT. On the high-pressure shaft system, an 8-stage HPC is in turn driven by a two-stage HPT. Figure 22d shows the final 3-D CAD model of the generic turbofan engine.

Military Application

An example of a highly-integrated propulsion system is shown in Fig. 23. As part of a multidisciplinary design process of a generic UCAV configuration, a turbofan engine was designed [32]. The engine geometry based on the presented central data model and engine performance data were utilized as boundary conditions for a CFD simulation which was part of an optimization process chain to determine the inlet duct geometry with low total pressure loss and low engine visibility.

CONCLUSIONS

In the context of this paper, an approach of a parametric data model for collaborative preliminary aircraft engine design was presented. The implementation is divided into the specific modeling and definition of aircraft engine components utilizing the modeling language UML and the software technical adaptation of the data structures by a C++ API.

The geometric parameterization and data modeling of relevant components of the aircraft engine system as well as the relationships between the components were highlighted. This digital representation allows for an efficient management of the intricate data flows during the collaborative preliminary design and thus helps to handle the extensive amount of transferred data between different disciplines and fidelity levels in a standardized manner. The highly flexible and abstract methodology enables the system to be seamlessly extended by additional engine components in future research activities.

The developed C++ API provides a variety of predefined functions to access the datasets for browsing, reading and writing. A tool or workflow from the collaborative design process receives input data via predefined interfaces and feeds back the generated output after successful execution. This is associated with a considerable potential for improvement in terms of error prevention and time optimization in the exchange of input and output quantities. All data model objects accessed in a process and all related modifications to the data are automatically monitored by the integrated provenance data management system.

The functionality of the approach was demonstrated by means of examples derived from applications in collaborative

preliminary engine design projects of DLR. It was shown that complex geometric characteristics and relationships on preliminary design level can be captured by the parametric definitions. The ability to provide characteristics with different levels of detail enables both multi-fidelity analysis and seamless transition between the different stages of engine design.

Although the approach was developed for collaborative preliminary engine design, the general functionality to support CFD processes has already been demonstrated [33–35]. The extension of the data model classes and the parametric definitions of the components in order to represent geometric properties on high-fidelity level are the next objectives and currently being addressed in DLR's internal projects SimBaCon (Simulation Based Certification) and VirTriP for the digitalization of the aero engine and stationary gas turbine sector.

Acknowledgments

The authors would like to thank the GTlab development team, the associates from the Institute of Propulsion Technology, Structures and Design, Simulation and Software Technology, Air Transportations Systems, and the entire PEGASUS and PERFECT project team for the help and collaboration enabling the present study.

References

[1] Jones, M., Bradbrook, S., and Nurney, K., "A Preliminary Engine Design Process for an Affordable Capability," Tech. Rep. No. ADP014191, Rolls-Royce PLC, Bristol, United Kingdom, 2003.

[2] Briceno, S. I., Laughlin, M. B., and Mavris, D., "A Virtual Simulation Platform for the Design, Testing, and Verification of Unmanned Aerial Vehicle Designs," *Simulia Community Conference*, 2014.

[3] Reitenbach, S., Krumme, A., Behrendt, T., Schnös, M., Schmidt, T., Hönig, S., Mischke, R., and Mörland, E., "Design and Application of a Multidisciplinary Predesign Process for Novel Engine Concepts," *Journal of Engineering for Gas Turbines and Power*, Vol. 141, No. 1, 2019.

[4] Denney, R., Tai, J., and Mavris, D., "Parametric Geometry for Propulsion-Airframe Integration," *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2012, p. 548.

[5] Denney, R., Gladin, J., Tai, J., and Mavris, D., "Propulsion Systems Modeling with Vehicle Sketch Pad," *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2013, p. 224.

[6] Hahn, A., "Vehicle sketch pad: a parametric geometry modeler for conceptual aircraft design," *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2010, p. 657.

[7] Goulos, I., Stankowski, T., Otter, J., MacManus, D., Grech, N., and Sheaf, C., "Aerodynamic design of separate-jet exhausts for future civil aero-engines—Part I: Parametric geometry definition and computational fluid dynamics approach," *Journal of Engineering for Gas Turbines and Power*, Vol. 138, No. 8, 2016.

[8] Heidebrecht, A., Stańkowski, T., and MacManus, D., "Parametric geometry and CFD process for turbofan nacelles," *Turbo Expo: Power for Land, Sea, and Air*, Vol. 49682, American Society of Mechanical Engineers, 2016, p. V001T01A033.

[9] Kulfan, B. M., "Universal parametric geometry representation method," *Journal of aircraft*, Vol. 45, No. 1, 2008, pp. 142–158.

[10] Böhnke, D., Nagel, B., and Gollnick, V., "An Approach to Multi-fidelity in Conceptual Aircraft Design in Distributed Design Environments," *IEEE Aerospace Conference*, 2011.

[11] Nagel, B., Böhnke, D., Gollnick, V., Schmollgruber, P., Rizzi, A., La Rocca, G., and Alonso, J. J., "Communication in Aircraft Design: Can we Establish a Common Language," *28th International Congress Of The Aeronautical Sciences*, 2012.

[12] Rumbaugh, J., Jacobson, I., and Booch, G., *The Unifed Modeling Language Reference Manual*, Addison-Wesley Object Technology Series, Addison-Wesley, Boston, MA, USA, 1999.

[13] Object Management Group (OMG), "Unified Modeling Language," `https://www.uml.org/`, 2019. Accessed: 2020-12-01.

[14] Open CASCADE, "Open CASCADE Technology, 3D Modeling and Numerical Simulation," `https://www.opencascade.com`, 2019. Accessed: 2020-12-01.

[15] Pratt, M. J., "Introduction to ISO 10303 — The STEP Standard for Product Data Exchange," *Journal of Computing and Information Science in Engineering*, Vol. 1, No. 1, 2001, pp. 102–103.

[16] Piegl, L. A., and Tiller, W., "Curve Interpolation with Arbitrary End Derivatives," *Engineering with Computers*, Vol. 16, 2000, pp. 73–79.

[17] Voss, C., and Nicke, E., "Automatische Optimierung von Verdichterstufen," *AG Turbo COOREFF-T FKZ*, 2008. Fachlicher Abschlussbericht Forschungsvorhaben, FKZ:0327713B, AG Turbo COOREFF-T.

[18] Bretschneider, S., "Knowledge-Based Preliminary Design of Aero-Engine Gas-Generators," , 2011. Dissertation.de.

[19] Sawyer, J., *Sawyers's Gas turbine Engineering Handbook*, 3rd ed., Vol. 1, Turbomachinery Interantional Publications, Norwalk, USA, 1985.

[20] DIN3990-1: Calculation of load capacity of cylindrical gears: introduction and general influence factors, "DIN," , 1987.

[21] Abbott, I. H., "Summary of Airfoil Data," Tech. Rep. NACA-TR-824, NASA (National Aeronautics and Space Administration), 1945. URL `https://ntrs.nasa.gov/search.jsp?R=19930090976`.

[22] Maeda, K., "Comparative survey of object serialization techniques and the programming supports," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, Vol. 5, No. 12, 2011, pp. 1488–1493.

[23] Leach, P., Mealling, M., and Salz, R., "RFC 4122: A Universally Unique Identifier (UUID) URN Namespace," 2005. Proposed Standard.

[24] 180-2, F., "Secure Hash Standard, Federal Information Processing Standard (FIPS), Publication 180-2," 2002. National Institute of Standards and Technology, US Department of Commerce, Washington D.C.

[25] Reitenbach, S., Vieweg, M., Hollmann, C., and Becker, R. G., "Usage of Data Provenance Models in Collaborative Multidisciplinary Aero-Engine Design," *Journal of Engineering for Gas Turbines and Power*, Vol. 142, No. 10, 2020. doi:10.1115/1.4048436, URL `https://doi.org/10.1115/1.4048436`, 101006.

[26] Schnell, R., Ebel, P.-B., Becker, R.-G., and Schoenweitz, D., "Performance analysis of the integrated V2527-engine fan at ground operation," 2013.

[27] Wolters, F., Becker, R., Schnell, R., and Ebel, P.-B., "Engine Performance Simulation of the Integrated V2527 - Engine Fan," *54th AIAA Aerospace Sciences Meeting, AIAA SciTech Forum*, 2016.

[28] Reitenbach, S., Becker, R., Hollmann, C., Wolters, F., Vieweg, M., Schmeink, J., Otten, T., and Siggel, M., "Collaborative Aircraft Engine Preliminary Design using a Virtual Engine Platform, Part A: Architecture and Methodology," *AIAA SciTech Forum*, 2020.

[29] Lengyel-Kampmann, T., Otten, T., Schmidt, T., and Nicke, E., "Optimization of an Engine With a Gear Driven Counter Rotating Fan—Part I: Fan Performance and Design," *Proceedings of the 22nd International Symposium on Air Breathing Engines, Phoenix, AZ, USA*, 2015, pp. 25–30.

[30] Otten, T., Lengyel, T., Becker, R.-G., and Reitenbach, S., "Optimization of an Engine with a gear driven counter rotating fan PART II: Cycle selection and Performance," 2015.

[31] Vieweg, M., Hollmann, C., Reitenbach, S., Schnoes, M., Behrendt, T., and Krumme, A., "Collaborative Aircraft Engine Preliminary Design using a Virtual Engine Platform, Part B: Application," *AIAA SciTech Forum*, 2020.

[32] Zenkner, S., Trost, M., Becker, R., and Voß, C., "Preliminary engine design and inlet optimization of the MULDICON concept," *Aerospace Science and Technology*, Vol. 93, 2019, p. 105318.

[33] Klein, C., Reitenbach, S., Schoenweitz, D., and Wolters, F., "A Fully Coupled Approach for the Integration of 3D-CFD Component Simulation in Overall Engine Performance Analysis," *Proceedings of ASME Turbo Expo 2017: Turbine Technical Conference and Exposition*, 2017. ASME Paper No. GT2017-63591.

[34] Klein, C., Wolters, F., Reitenbach, S., and Schönweitz, D., "Integration of 3D-CFD Component Simulation Into Overall Engine Performance Analysis for Engine Condition Monitoring Purposes," *Proceedings of ASME Turbo Expo 2018: Turbine Technical Conference and Exposition*, 2018. ASME Paper No. GT2018-75719.

[35] Hollmann, C., Mennicken, M., Otten, T., Schönweitz, D., and Wolters, F., "Evaluation of Boundary Layer Ingestion on Propulsion Level by Coupling of Overall System Analysis and High-Fidelity 3D-CFD Fan Simulation," 2019.