DESIGN OF A COMMON LIBRARY TO SIMPLIFY THE IMPLEMENTATION OF AIRCRAFT STUDIES IN CPACS

J. Jepsen, P.-D. Ciampa, B. Nagel, V. Gollnick,
Deutsches Zentrum für Luft- und Raumfahrt (DLR), Lufttransportsysteme (LY),
Blohmstraße 18, 21079 Hamburg, Deutschland

Abstract

Increasing the degree of automation in the aircraft design process is a necessary step in order to find the aircraft concept of the future. Due to the manifold of requirements, expert knowledge from several disciplines is required. Each discipline uses its own concepts to reduce the system complexity and to focus on the properties important for the specific problem. Those concepts allow an efficient description of the problem and possible solutions. A central data format such as CPACS can never combine all the concepts needed for all the different disciplines. In this paper a library is presented which supports the definition and implementation of engineering concepts and thus reduces the effort for automating the aircraft design process.

1. INTRODUCTION

The air transportation system is facing challenges to reduce the climate impact, while forecasts predict an increase in air traffic by a factor of 6 until 2050 [1]. Achieving the ambitious goals set by the ACARE for the year 2050, such as reducing the CO2 emissions by 75% and reducing the NOx emissions by 90% per passenger kilometer compared to the year 2000, requires a major effort to improve the efficiency of the entire air transportation system. One way of reducing the environmental emissions is aircraft design where a decrease in fuel consumption directly effects the polluting emissions. There are different promising unconventional aircraft concepts under investigation, and a multi-disciplinary analysis is necessary to realistically estimate their potential. Therefore collaboration between experts from different disciplines is becoming more and more important. Current research at DLR hence involves the development of collaboration methods to create a fruitful environment for a holistic aircraft design process [2], [3].

In order to find a design which optimally satisfies all the requirements, a huge design space needs to be searched. Designing and analyzing a large number of different aircraft designs in a reasonable time can only be achieved through an efficient process with a high degree of automation. This especially counts for unconventional designs which, due to the missing statistical database, need to be investigated with physically based methods. These methods usually require a more detailed model than commonly available in the early design stages. Model generators such as openVSP and the MMG automate the design process and support the designer in the initial definition of an aircraft as well as with the creation of the different disciplinary models [4]–[7]. These model generators already need to consider all the relevant aspects/disciplines of the aircraft design process to generate a feasible design. Therefore knowledge coming from the different disciplines needs to be implemented into the model generators. This requires capturing, structuring and formalizing design knowledge.

Recently the trend is towards a central model from which all the disciplinary models are generated. While for the MMG and openVSP the central model is implemented within the software which supports the generation of all the different disciplinary models, the approach followed by DLR uses the central data exchange format CPACS. The XML based CPACS format seems to be presently successful due to its simplicity and flexibility. It is used in national and international projects [8], [9].

Finding/Developing a central data format in which all the information from the different disciplines can be stored equally efficient is unlikely ever to succeed. One can describe the search for a central data format as the task to find a knowledge representation for the aircraft design process which serves all the disciplines as good as possible. But this mission is bound to fail. The efficiency with which problems and their solutions can be described is strongly depending on the choice of knowledge representation as described in [10]. That is due to the nature of how the human mind processes information. Research on human intelligence confirms that the human brain uses different layers of abstraction in order to deal with otherwise too much and too complex information [11]. Abstract concepts help reducing the complexity of a system by putting focus on the important parts, ignoring all details not relevant for the problem solving process. In the aircraft design process a large number of concepts are used to breakdown the problem into smaller, less complex pieces. Depending on the scientific field/discipline different concepts are used. A central data format such as CPACS can never inhabit all the concepts without redundancy and tremendous overhead. Therefore compromises have to be made for the development of the central data format. But since concepts are an important part in the knowledge management process, a way to support different concepts need to be provided.

For an efficient use of a central data format users should be able to access the data through an interface which supports the discipline specific/problem specific concepts. In this paper we will present the design of a library which supports the implementation of concepts for a high level access to CPACS data. By using this library the development time of an aircraft design process can be reduced by enhancing the following two phases:

1. Supporting the definition of new concepts during the concept definition phase

2. Supporting the reduction of the system complexity through simplification and decomposition

The following section starts with a short introduction to the cpacsPy library and its structure. We present two libraries which are used to access CPACS data as well as a place for implementing extension modules such as concepts. Two packages and their modules which are included in the cpacsPy library are shortly introduced.

In section 3, a simplified definition of the Wing concept included in the cpacsPy library is given, followed by the definition of the Boxwing concept and its properties.

After the concepts are defined the steps for implementing it for the CPACS data format are described in section 4. At first the necessary steps without the benefit of having the cpacsPy library are discussed, followed by the process making use of the cpacsPy. The comparison of the two illustrates the potential of reduced development time when using it.

In order to show the flexibility of the implemented Boxwing concept, in section 5 a parametric study on an aircraft design process is presented. Within the process different analysis methods are used using models generated from a CPACS file which was changed using the Boxwing concept.

Section 6 wraps up the content of the paper and gives some indication for future development.

2. LIBRARY FOR THE MODULAR IMPLEMENTATION OF CONCEPTS FOR CPACS

As mentioned in the introduction, the human mind uses concepts to filter information. The central data format CPACS is a big information resource which needs to be filtered. The cpacsPy library supports the implementation of modular concepts to provide parameters that are not part of the CPACS data format but which need to be composed from several pieces of information.

Engineering knowledge can be described in a lot of different ways. But when there is a restriction on the expressions that can be used it becomes much more difficult. Formalizing knowledge without being able to use the proper concepts is as if writing in English without using the letters “e” and “t”. It can be done but only with tremendous effort. By using the cpacsPy library the efficiency for capturing and implementing design knowledge can be increased. Through the library the engineering knowledge can be linked to the CPACS data format and thus directly be used for design tasks in different disciplines, thus reducing the development time of new design methods.

The cpacsPy library is implemented in Python. Python is a high level programming language with a large standard library and widespread in the scientific community [12], [13]. It can be used for rapid development of prototypes as well as for the implementation of production code which makes it well suited for scientific applications.

The development is conducted using the test driven development technique to ensure software quality. Test driven development describes a development process where the automated tests for a piece of software are written before the actual feature/code is implemented [14]. During the development of the cpacsPy library object oriented design methods are used to structure the software and allow a simple extension through class inheritance [15].

cpacsPy closely integrates the TIXI and TIGL libraries. TIXI is an XML interface which includes some CPACS specific features such as generating, reading and updating basic CPACS node types. TIGL is a geometry library for the CPACS data format. It uses the OpenCascade library to construct 3-dimensional shapes which for instance can be used to calculate surface points [16]. TIGL requires TIXI to access CPACS files. Both, TIXI and TIGL, are developed at DLR Simulation and Software Technology.

In FIGURE 1 the overall structure of the cpacsPy library is shown.

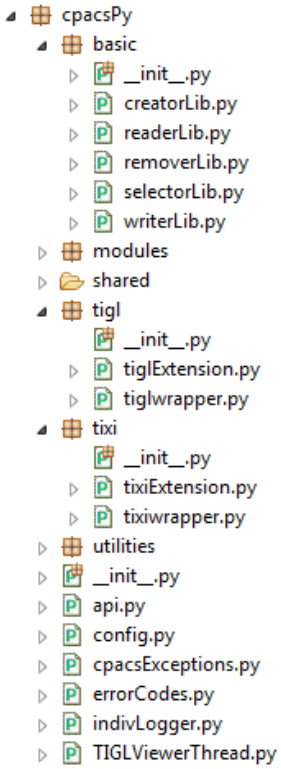

FIGURE 1. cpacsPy structure

For the access to CPACS files it implements high level functions using TIXI and TIGL.

All core functions, such as reading, writing, selecting, creating and deleting basic CPACS data are implemented in the basic package. The tixi and tigl packages are used for the integration and extension of the TIXI and TIGL libraries. The tixiwrapper and tiglwrapper for Python are

provided with the TIXI and TIGL libraries. The tixiExtension and tiglExtension modules contain subclasses of the Tixi- and Tigl-classes respectively for the implementation of missing features[1].

The main part of the library is made up by the modules package where all the concept modules are placed. The concepts carry the main knowledge on the different views on the CPACS data. Currently it contains concepts for the aerodynamic shape of wings and fuselages. The "wingAeroShape" package includes a model description of a wing consisting of a wing component, the wing segments and the wing elements which link to the different airfoils.

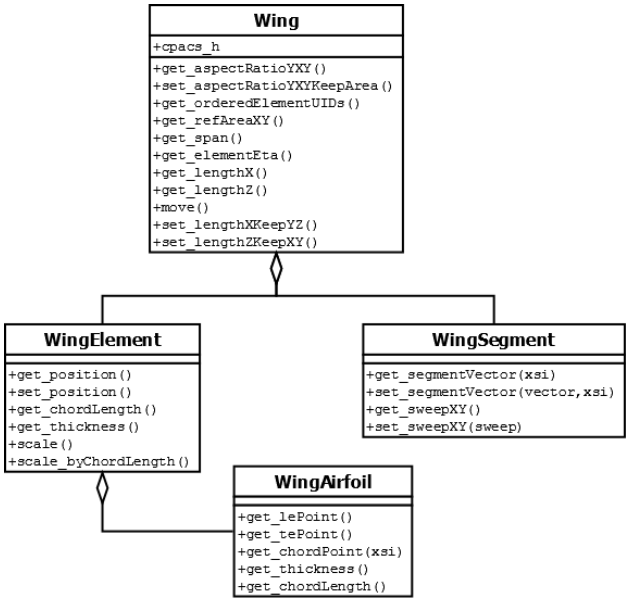

FIGURE 2. Wing Class

Likewise the "fuselageAeroShape" package includes analog modules for the fuselage. These modules implement concepts which are typically used in the conceptual and preliminary aircraft design.

Different concepts can be useful for describing and solving different problems. User software which uses the cpacsPy library can rely on concepts whose interfaces are independent of the data format behind it[2]. Accordingly the knowledge and rules of the user software should not include data format specific content/parameters but be consistent with the concept they use/describe.

When implementing a new concept for the cpacsPy library, users need to follow some guidelines to ensure a smooth integration into the cpacsPy environment. How to implement a concept for the cpacsPy library is described in section 4.2. But first the concept of a Boxwing is defined in the next section.

[1] This should be used for an easy implementation of missing features. The features can be implemented in the TIXI/TIGL library at a later time.

[2] In the cpacsPy library the concepts are only implemented for the CPACS data format.

3. DEVELOPING A CONCEPT OF A BOXWING

This section describes the engineering concept of a Boxwing used to perform a parametric study in a conceptual aircraft design process. As a simplification this concept concentrates on the planform definition. By referring to an engineering concept of a conventional wing, which is available in the cpacsPy library, the structural complexity of the Boxwing, as defined by [17], can be reduced (decomposition). Although the concept is kept simple it can be applied to a detailed geometry without losing more detailed information. The Boxwing used in this paper is shown in FIGURE 3.

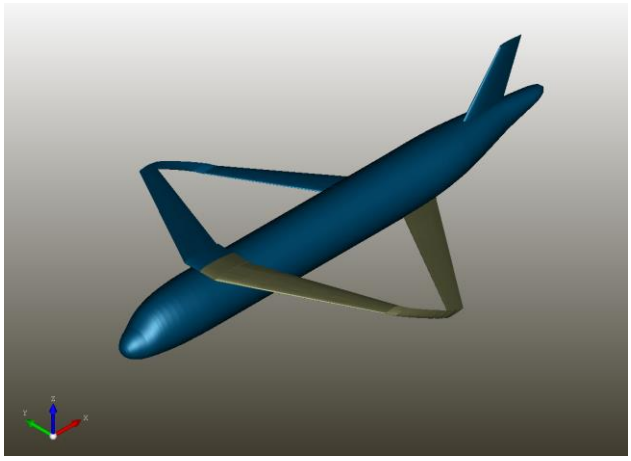

FIGURE 3. Boxwing Configuration

The concept describes the geometry by three separate wings, the top wing, the bottom wing and the vertical wing which connects the top wing with the bottom wing. In the next section the definition of the single wing concept is shown.

3.1. The Wing Concept

While a thorough definition of all the details of a wing are required in the final design stage, in conceptual design a rough description of the geometry is sufficient. For the sake of simplicity, only those properties of the wing concept, implemented in the cpacsPy, which are used for the Boxwing definition, are shown. This includes the following properties[3]:

- Taper Ratio
- X-Length
- Y-Length (Span)
- Z-Length
- Reference Area
- Aspect Ratio

Each wing fitting in this concept should have at least an airfoil description at the root and one at the tip of the wing. With the chord length of the root and tip, the taper ratio is defined as in (1):

$$\tau = \frac{c_{tip}}{c_{root}} \quad (1)$$

[3] Since, in this paper, we concentrate on the wings planform, the twist distribution is not part of this particular concept.

The definition of the span requires a bit more thought. While usually being described as the distance between root and tip along a reference axis, this might not always be correct. For a C-wing as shown in FIGURE 4, that definition does not refer to the intended parameter as described by “a + b”. A more precise definition for the span is the sum of all change in y-axis along the wings reference line. E.g. either the leading edge or the quarter chord line is used as the reference line (y-axis). This definition can also deal with the C-wing shape[4]. The x-length and z-length are defined accordingly along their respective axis.

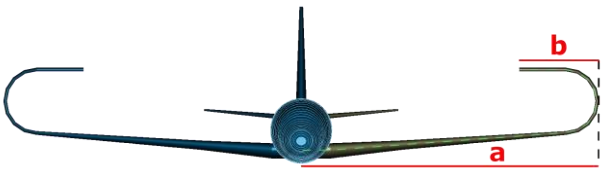

FIGURE 4. C-Wing configuration

The same approach is used for the definition of the reference area. It should also be applicable to a C-wing shape which leads to the following definition: The reference area is the area resulting from an integration of the chord length projected on the reference plane, measured along the same reference axis as the span definition.

The aspect ratio describes the stretching of the wing by using the wing span and reference area. Equation (2) gives the definition of the aspect ratio.

$$(2) \quad \Lambda = \frac{b_f{}^2}{S_{ref}}$$

With only these four parameters the planform of a wing can be described. In the next section this wing concept is used to describe the properties of a Boxwing.

3.2. The Boxwing Concept

For the Boxwing concept we define the following three properties:

- Y-Length (Span)
- Reference Area
- Aspect Ratio

Since this Boxwing concept is based on the three wings approach these properties are defined using the properties of the single wing concept for each of the wings. The geometric properties are mainly driven by the top and bottom wings geometry. Therefore the influence of the vertical wing is neglected for the described concept. Thus the span of the Boxwing can be defined as the sum of the top wing span and the bottom wing span as shown in equation (3).

$$(3) \quad b_{f,boxwing} = b_{f,top} + b_{f,bottom}$$

The reference area of the Boxwing is defined analogous as shown in (4).

$$(4) \quad S_{ref,boxwing} = S_{ref,top} + S_{ref,bottom}$$

It can be argued what would be a sound definition for the taper ratio of the Boxwing concept. Since there is no single root and tip chord the classical definition fails. For the Boxwing concept developed in this paper, the mean value of the top and bottom wings taper ratio is used as described by equation (5).

$$(5) \quad \tau_{boxwing} = \frac{\tau_{top} + \tau_{bottom}}{2}$$

Using equation (2) from the single wing concept the aspect ratio of the Boxwing can be defined as in (6).

$$(6) \quad \Lambda_{boxwing} = \frac{(b_{f,top} + b_{f,bottom})^2}{S_{ref,top} + S_{ref,bottom}}$$

As a special case when the spans of both wings are the same the equation can be simplified to (7).

$$(7) \quad \Lambda_{boxwing} = \frac{(2 \cdot b_f)^2}{S_{ref,top} + S_{ref,bottom}}$$

For Boxwing aircraft the top and bottom wings usually have the same reference area and span resulting in an identical aspect ratio. The equation in (6) allows the calculation of the aspect ratio even if the spans of the top and bottom wings differ. Although, in that case the span of the vertical wing should no longer be neglected but included in the calculation of the aspect ratio. Anyhow the configuration used in this paper is limited to configurations with the same span for both wings.

A change in aspect ratio can be realized using different boundary conditions. Since the aspect ratio is a dimensionless parameter, changing it should only affect the shape but not the size of the wing. Therefore the first boundary condition is to change the aspect ratio while keeping the reference area constant. From equation (7) it can be seen that in order to do that, one needs to allow the wings span to be changed. For wings with different spans the change in span should be equally distributed between the two wings with respect to their percentage in span. The resulting ratios as shown in (8) are therefore kept constant.

$$(8) \quad 2 = \frac{b_{f,top}}{b_f} + \frac{b_{f,bottom}}{b_f}$$

By solving equation (7) for b_f the required span can be calculated and distributed between the two wings by (9) and (10).

[4] From an aerodynamic point of view this definition is more reasonable, but when talking about airport slots a different definition of the span is needed. This illustrates how depending on the context parameters can be interpreted in different ways. Hence making the use of a proper knowledge management indispensable for a highly collaborative process.

(9) $$\hat{b}_{f,top} = \frac{\bar{b}_{f,top}}{\bar{b}_f} \cdot \hat{b}_f$$

(10) $$\hat{b}_{f,bottom} = \frac{\bar{b}_{f,bottom}}{\bar{b}_f} \cdot \hat{b}_f$$

Where the index $\bar{b}_f$ marks the original values and $\hat{b}_f$ marks the updated/changed values.

While increasing the span the average chord length of the wing needs to be decreased in order to keep the reference area constant. The change in the average chord length can be achieved by different means. Therefore another boundary condition is needed to define the way the chord lengths are changed. As the default case for the Boxwing concept it was decided that the taper ratio is kept constant. Hence the chord lengths are to be changed equally with respect to their original size. In general concepts might implement different methods to change their properties which can be useful for different applications.

After this procedure has been performed for the top and bottom wing the vertical wing needs to be adjusted to connect the top with the bottom wing. For the Boxwing concept it was decided to adapt the vertical wing in the following way:

1) The span (y-length) of the vertical wing, as defined in section 3.1, is constant.
2) The chord lengths are changed with the same factor as used for the top and bottom wings chord lengths.
3) The vertical wing is stretched along the x- and z-axis in order to connect with the top and bottom wing.

With this definition of a Boxwing concept and the default behavior for changing its aspect ratio, a design study can be performed using the high-level parameter of the aspect ratio as a design parameter. The structural complexity of the Boxwing definition was reduced by referring to the existing concept of a wing, consequently reducing the effort for the definition process. In the following section we show how the implementation effort can be reduced by using the cpacsPy library. In the following section we show how the cpacsPy library can be used to save time during the implementation of the concept for CPACS. Therefore we give an impression of the implementation effort required with directly using CPACS first, followed by the implementation steps necessary when using cpacsPy.

4. IMPLEMENTATION OF PARAMETRIC CHANGES OF A BOXWING

During the setup of a design study one needs to consider how to perform the design changes which are to be analyzed. Although CPACS has been used for design studies in several research projects there is currently no simple way to perform high-level geometric changes. The following section demonstrates the current practice for the implementation of design studies in CPACS.

4.1. Current practice for design studies in CPACS

There are basically two common approaches to create variations of CPACS geometry for design studies.

The first approach uses a program which contains a parametric description of the configuration which is analyzed. That program generates the aircraft geometry in CPACS from scratch and is usually tailored to a specific configuration. It works well if the created geometry is always the first to be created and all other data can be added afterwards. This approach was chosen for the design study in [18]. But when attempting to use an existing CPACS file, which might also include additional data, this approach cannot be used. The second approach and the one chosen for this paper is to perform changes by manipulating the CPACS definition of an existing CPACS file. Currently TIGL is not capable of performing changes to the aircraft geometry. For an aircraft designer to change the aerodynamic shape of a wing he/she must change the CPACS definition of the geometry.

In CPACS geometric shapes can be described in multiple ways, for instance by explicitly defining Cartesian points in the Euclidean space, and eventually by applying affine transformations with respect to several coordinate systems (e.g. wing based, and airfoil based coordinate systems) [19]. On the one hand the flexibility offered allows to represent complex shaped components, on the other hand geometrical aggregative parameters often used in the pre-design stages, such as wing aspect ratio and wing span, are not explicitly defined in the CPACS data format. More specific CPACS includes at least five coordinate systems which are setup in the following order starting with the inertial coordinate system, followed by the component coordinate system, then the section coordinate system which includes the element coordinate system which finally links to the coordinate system of the profile. The first four coordinate systems are shown in FIGURE 5. The profile coordinate system, which is not shown, defines the profile points placed by the element.

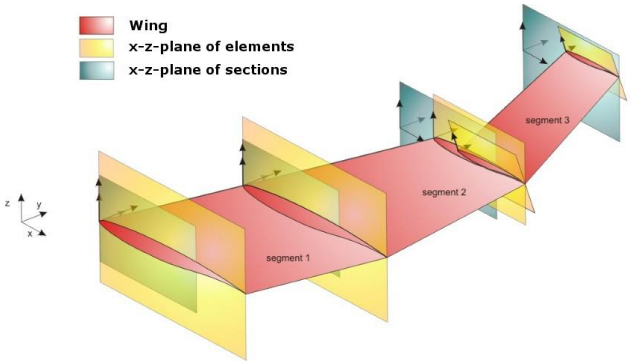

FIGURE 5. Coordinate systems in CPACS

Affine transformations can be performed in almost all of these coordinate systems resulting in a large number of possibilities to define a single geometry. When a user intends to allow all CPACS conformable data as input, a large number of cases need to be considered which makes the implementation rather complex.

In CPACS neither the aspect ratio, nor the span or the reference area are defined as parameters which is due to their conceptual nature. These parameters are part of a concept of a wing which can be simply described by a couple of parameters. But as a central data format for all kinds of aircraft CPACS is designed to hold all different kind of aircraft concepts, even those which might not be defined by such parameters for conventional aircraft. For example - How would you describe the taper ratio of a flying saucer? Parameters such as the aspect ratio or taper ratio belong to a concept of an aircraft and can only be applied to aircraft which fit to that concept.

When working close to a specific data format, the way of thinking about problems is always influenced by the structure or knowledge representation of the format. In CPACS each wing component is setup by an arbitrary number of linear segments. This structure already influences the way in which possible solutions are searched for. While the change in aspect ratio for the Boxwing concept in section 3.2 could be described rather simple, the translation for an implementation close to the CPACS data format requires much more effort. It needs to be considered how transformations in the different coordinate systems affect the way in which a desired change of the geometry can be achieved.

Although the use of a central data format is necessary, the restrictions due to the chosen structure for the data format which can be seen as a particular knowledge representation cannot be denied. In order to counter these restrictions the use of a central data format should be supported by a modular and flexible interface which supports the implementation of concepts. The cpacsPy library presented in section 2 is designed as an interface to implement concepts through modular extensions making use of object oriented design techniques. In the following section we show how the cpacsPy can be used and extended to realize a design study of the aspect ratio applied to a Boxwing aircraft.

4.2. Implementing a Boxwing concept with cpacsPy

Writing a concept of a Boxwing for the cpacsPy library enables a simple reuse and extension of the code at a later time also by other researchers. Before a new concept can be implemented a decision has to be made on how the concept should be categorized. Since the concept described in this paper considers the aerodynamic shape of the Boxwing, it will be placed in a package called "aeroShapeBoxWing". Therefore a new Python package with that name is created in the modules package of cpacsPy. Inside it a new module, named "BoxWingLib", is created which implements the "BoxWing" class. FIGURE 6 shows the layout of the “BoxWing” class and the classes it uses.

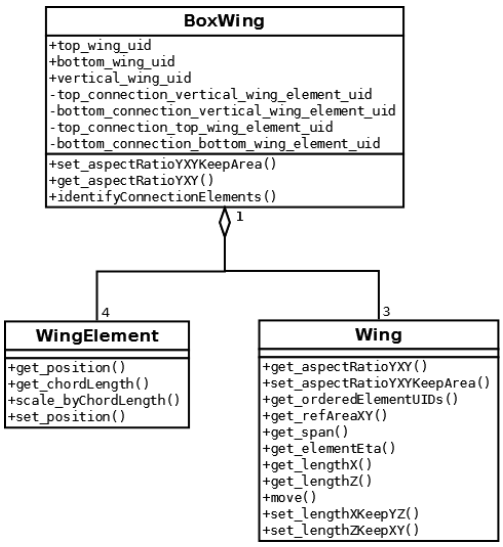

FIGURE 6. BoxWing class and its aggregates

Every concept needs to keep a handle of the cpacsPy instance to access the CPACS data which is represented by the "cpacs_h" attribute. The "BoxWing" concept is built by using an assembly of three "Wing" concepts from the “wingAeroShape.wingLib” module. Hence the unique identifiers (UID) for the three wings (top_wing_uid, bottom_wing_uid, vertical_wing_uid) are also saved as class attributes. With the three UIDs the calculation of the aspect ratio of the Boxwing can be implemented.

Using the existing "Wing" concept allows a high level implementation which is close to the conceptual knowledge. From the code in FIGURE 7 the main steps can be clearly identified.

```
1  def get_aspectRatioYXY(self):
2          top_wing_xpath = self.createWingPath(self.top_wing_uid)
3          bottom_wing_xpath = self.createWingPath(self.bottom_wing_uid)
4
5          top_refArea = self.get_refAreaXY(top_wing_xpath)
6          top_span = self.get_span(top_wing_xpath)
7          bottom_refArea = self.get_refAreaXY(bottom_wing_xpath)
8          bottom_span = self.get_span(bottom_wing_xpath)
9          return (top_span + bottom_span)**2 / (top_refArea + bottom_refArea)
```

FIGURE 7. Calculating the Aspect Ratio

In the first two lines the xpaths to the two wing components are created. The lines 5 and 7 calculate the reference area (the area of the wings projection on the x-y-plane) of the two wing components. Lines 6 and 8 calculate the span (length of the wings projection on the y-axis) of the two wings. In line 9, equation (7) for calculating the Boxwings aspect ratio is implemented and the result is returned. This short example shows that by using the right concepts, knowledge can be implemented in a way close to the problem domain[5].

Changing the aspect ratio of the top and bottom wing is performed by simply calling the respective method of the single Wing concept for each of the wings. Since the aspect ratio of both wings is kept the same for all cases, the aspect ratio for each wing can be calculated by halving the Boxwings aspect ratio as shown in (11).

[5] Similar to a Domain Specific Language (DSL).

(11) $$\Lambda_{top|bottom} = \frac{\Lambda_{boxwing}}{2}$$

But applying the changes for the connection wing requires a bit more effort. At first the wing tips need to be identified. The interface to the wing tips is provided by the "WingElement" concept. As shown in FIGURE 6 it provides methods to change the shape of a wing cross section.

Next the x-length and z-length of the connection wing is adapted to the new distance between the wing tips of the top and bottom wing (with respect to their leading edge). Then the average chord length is changed by the same factor as for the top and bottom wing. Subsequent the connection wing is moved to the position of the bottom wing tip. Due to rounding, the tip of the top wing and the tip of the connection wing might not coincide completely. Therefore the position of the respective connection wing tip is "snapped" to the top wings wing tip.

Henceforth the implemented concept can be used to access Boxwings in CPACS which use the three wing approach for their shape definition. In the next section an aircraft design process is setup which uses the concept in order to show its robustness.

5. AIRCRAFT DESIGN PROCESS

Once the implementation of the Boxwing concept is finished, it can be tested in a common use case such as an aircraft design study. At DLR Air Transportation Systems, the implementation of aircraft design processes is usually done using the integration framework RCE (Remote Component Environment). In RCE different design and analysis tools can be integrated into a workflow [20], [21]. FIGURE 8 shows the workflow of an aircraft design process implemented at DLR.

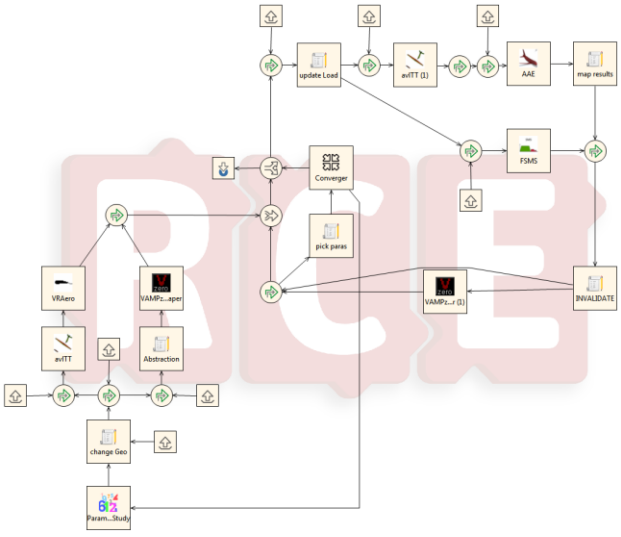

FIGURE 8. Aircraft design Workflow in RCE

The workflow contains the RCE component “Parametric Study” which is used to create the parameter variation of the Boxwings aspect ratio. The value is then applied to the CPACS file through the implementation of the concept as described in the previous section. For the used CPACS file, a single change of the aspect ratio results in a change of 153 CPACS parameters. Those 153 parameters reflect all the required changes as described in the concept definition of the Boxwing in section 3.2. Some exemplary designs are shown in FIGURE 9 to FIGURE 11.

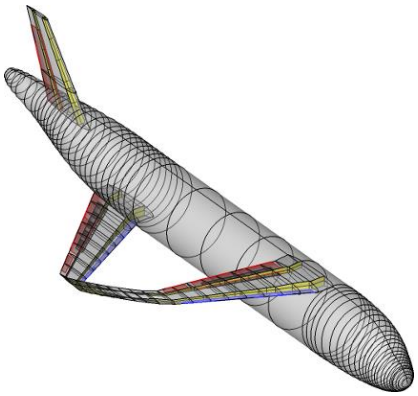

FIGURE 9. Boxwing design with an aspect ratio of 10

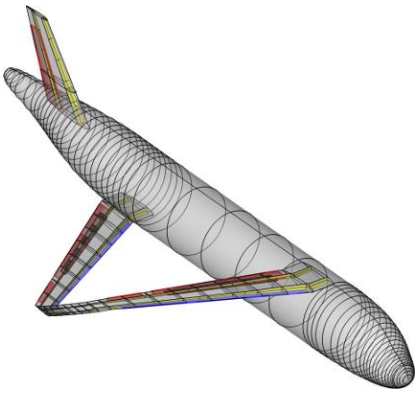

FIGURE 10. Boxwing design with an aspect ratio of 20

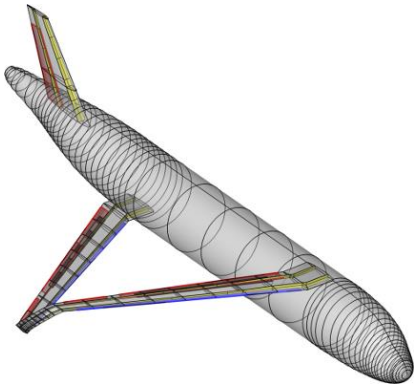

FIGURE 11. Boxwing design with an aspect ratio of 30

The figures also show the inner wing structure and control surfaces. This is due to the fact that in CPACS the inner wing structure as well as the control surfaces are defined with respect to the aerodynamic shape of the wing, thus they are scaled according to the outer shape. This demonstrates how a complex and relatively detailed model can be changed by using a simple engineering concept.

Within the aircraft design process the new CPACS aircraft model is used for the creation of the disciplinary analysis models. The quality of the geometry output allows the creation of models for the vortex lattice method as shown in FIGURE 12 to FIGURE 14.

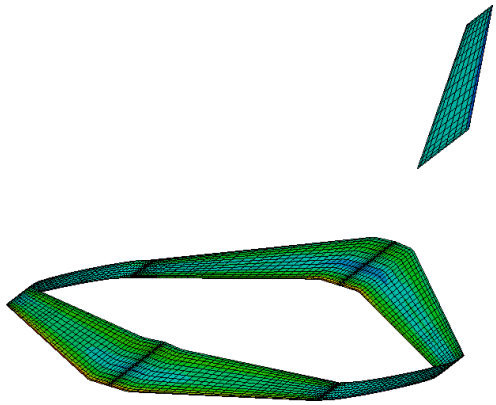

FIGURE 12. AVL model with an aspect ratio of 10 (c_p distribution)

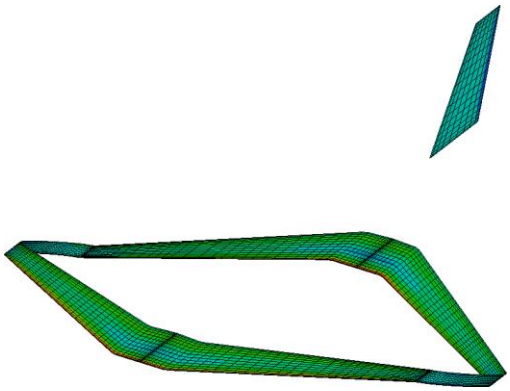

FIGURE 13. AVL model with an aspect ratio of 20 (c_p distribution)

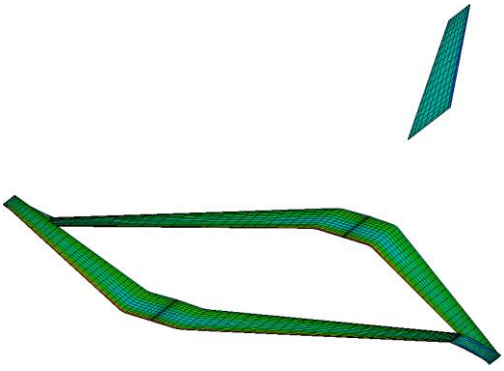

FIGURE 14. AVL model with an aspect ratio of 30 (c_p distribution)

The detailed structural model is used for the FEM analysis of the AAE tool which performs the wing sizing under consideration of loads calculated by the vortex lattice method in avlTT. Models of the wing box for the wing sizing are shown in FIGURE 15 to FIGURE 17. The aerodynamic performance and the engine performance, of which the latter was provided by DLR Propulsion Technology, is used to perform a mission simulation with FSMS. Finally VAMPzero is used to update the mass estimation including the masses for the fuel and wing calculated by the other analysis tools.

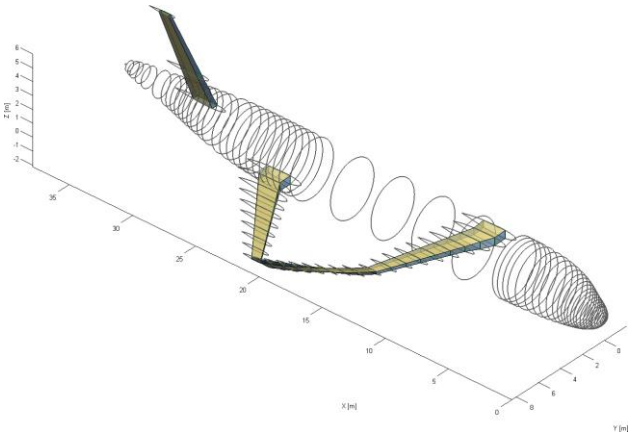

FIGURE 15. Wing box model (aspect ratio of 10)

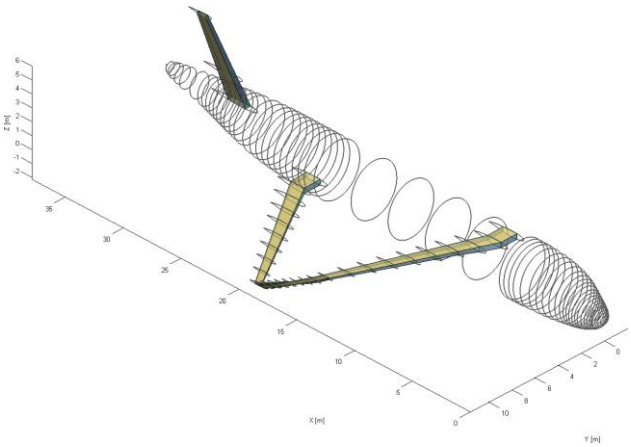

FIGURE 16. Wing box model (aspect ratio of 20)

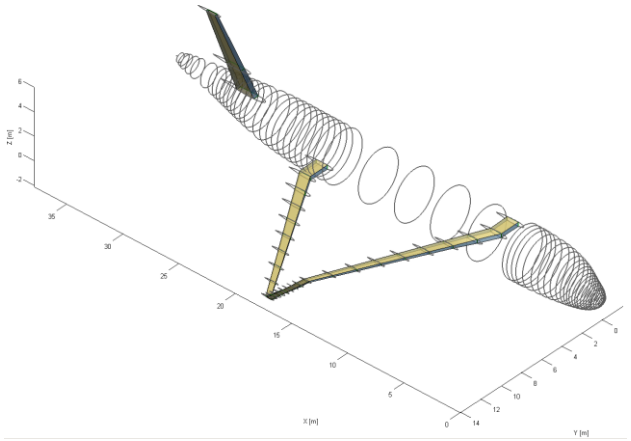

FIGURE 17. Wing box model (aspect ratio of 30)

The focus of the performed study was to test the reliability of the geometric changes performed by the implemented Boxwing concept. An extensive analysis of the presented Boxwing configuration can be found in [18].

6. CONCLUSION

In this paper a definition for a Boxwing concept was presented. By using the decomposition principle, the Boxwing could be described three wing components. The example of the Boxwing concept illustrates how comparably simple the definition and implementation of a new concept can be, by referring to the cpacsPy's wing concept. The complexity of changes in the data format is hidden through the use of the aspect ratio property. In the implemented concept a single change of the aspect ratio results in 153 changes in CPACS parameters. The fact that a more detailed geometry can be changed through a simplified concept enables the combination of methods from different levels of fidelity into a single aircraft design process. Of course the presented Boxwing concept can be extended to include other parameters or a more general concept could be defined which is not restricted to the three wing approach but which can deal also with other constructs. Adding more and more concepts to the cpacsPy library increases the possibilities in which disciplinary engineering knowledge can be expressed. Thus preparing the basis for a knowledge based model generator using the CPACS data format.

Thus the presented cpacsPy library creates a bridge between engineering parameters (defined in concepts), and the product parameters specific to the chosen data format (in this case CPACS). By pushing a formal definition of disciplinary concepts, problem specific knowledge can be described and implemented more efficient. The potential for reducing the development time was shown by the comparison of the common implementation process with the one supported by the cpacsPy.

The formalization of knowledge in a form processible by computers is an essential step towards a higher automation in aircraft design but also in general product design. Methodologies such as MOKA or KNOMAD intensively define processes for capturing, formalizing and implementing engineering knowledge. Such methodologies could be used for an exhaustive knowledge management in the future as shown by [4], [5]. The implementation of the cpacsPy library constitutes less change to the daily work of the engineer and is therefore more likely to be well-accepted, in an environment where every developer has its individual "working style". Once the cpacsPy is used and the benefits of a collective knowledge system are visible to each individual, a methodology like MOKA could be more easily established. Summing up, the cpacsPy supports aircraft designers in the definition and implementation of design knowledge. Once concepts from different disciplines are available, it also simplifies the implementation of cross disciplinary knowledge which is required in a collaborative multi-disciplinary design process. Using that knowledge to increase the degree of automation in an aircraft design process results in a higher efficiency, thus allowing the analysis of a larger design space. This finally raises the chances of finding an efficient aircraft design which contributes to fulfilling the ACARE goals for the air transportation system of the future.

References

[1] M. Dareck, C. Edelstenn, and T. Ender, “Flightpath 2050 Europes Vision for Aviation,” *… Off. Eur. …*, 2011.

[2] E. Moerland, T. Zill, B. Nagel, H. Spangenberg, H. Schumann, and P. Zamov, “Application of a distributed MDAO framework to the design of a short- to medium-range aircraft,” *61th Ger. Aerosp. Congr. (DLRK), Berlin, Ger.*, 2012.

[3] A. Bachmann, J. Lakemeier, and E. Moerland, “An Integrated Laboratory for Collaborative Design in the Air Transportation System,” pp. 1–12.

[4] L. Rocca, *Knowledge Based Engineering Techniques to Support Aircraft Design and Optimization*. 2011.

[5] G. La Rocca, “Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design,” *Adv. Eng. Informatics*, vol. 26, no. 2, pp. 159–179, 2012.

[6] W. J. Fredericks, K. R. Antcliff, G. Costa, N. Deshpande, M. D. Moore, E. A. S. Miguel, and A. N. Snyder, “Aircraft Conceptual Design Using Vehicle Sketch Pad,” 2010.

[7] A. Hahn, “Vehicle sketch pad: a parametric geometry modeler for conceptual aircraft design,” *48th AIAA Aerosp. Sci. Meet. Exhib.*, 2010.

[8] N. et al. Kroll, “DLR-Projekt Digital-X: Auf dem Weg zur virtuellen Flugzeugentwicklung und Flugerprobung auf Basis höherwertiger Verfahren,” 2014.

[9] B. Nagel, D. Böhnke, V. Gollnick, P. Schmollgruber, a. Rizzi, G. La Rocca, and J. J. Alonso, “Communication in Aircraft Design: Can We Establish a Common Language?,” *28th Int. Congr. Aeronaut. Sci.*, 2012.

[10] S. Russel and P. Norvig, “First-Order Logic,” in *Artificial Intelligence: A modern approach*, Upper Saddle River: Prentice Hall, 2010, pp. 285 – 321.

[11] J. Hawkins, *On Intelligence*. New York: St. Martin’s Griffin, 2004.

[12] P. Kaiser and J. Ernesti, *Python*. Galileo Computing, 2008.

[13] “Python Documentation.” [Online]. Available: www.python.org/doc/. [Accessed: 20-Jul-2015].

[14] B. K. Beck and P. Date, “Test-Driven Development By Example Test-Driven Development By Example,” 2002.

[15] B. Meyer, *Object-Oriented Software Construction*. Prentice Hall International (UK) Ltd, 1988.

[16] M. Siggel, “TIGL 2.1.6 API,” 2015. [Online]. Available: http://tigl.sourceforge.net/Doc/index.html. [Accessed: 25-Jul-2015].

[17] E. Stage and I. Comments, “The clios process,” pp. 1–12, 2007.

[18] T. Pfeiffer, E. Moerland, and V. Gollnick, “ANALYSIS OF AIRCRAFT CONFIGURATIONS INCLUDING PROPAGATED UNCERTAINTIES.”

[19] Deutsches Zentrum für Luft- unf Raumfahrt, “CPACS 2.2.1 Documentation.” 2015.

[20] M. Kunde and A. Schreiber, “Advantages of an Integrated Simulation Environment.”

[21] D. Seider, P. M. Fischer, M. Litz, a. Schreiber, and a. Gerndt, “Open source software framework for applications in aeronautics and space,” *2012 IEEE Aerosp. Conf.*, pp. 1–11, 2012.