# An Approach to Multi-fidelity in Conceptual Aircraft Design in Distributed Design Environments

Daniel Böhnke*, Björn Nagel, Volker Gollnick
Institute of Air Transportation Systems
German Aerospace Center (DLR e.V.)
Blohmstrasse 18, 21079 Hamburg, Germany

*Abstract*—In the present study we introduce a design environment consisting of a framework, a central model and a newly developed conceptual design module. The Common Parametric Aircraft Configuration Scheme (CPACS) is the standard syntax definition for the exchange of information within preliminary airplane design at DLR. Several higher fidelity analysis modules are already connected to CPACS, including aerodynamics, primary structures, mission analysis and climate impact. The analysis modules can be interfaced via a distributed framework. To initialize the design processes, capabilities are needed to close the gap between top-level requirements and preliminary design. Additionally, results of a design loop need to be merged to generate inputs for further iterations and convergence control. For this purpose we developed a conceptual design module based on handbook methods where the focus is set on multi-fidelity. For the upward change in level of detail a knowledge-based approach is used for the generation of CPACS models. This includes the geometry generation, as well as additional data such as the mass breakdown and the tool-specific inputs for further analyses in higher fidelity modules. The feedback loop is closed downwards by reducing the granularity from the CPACS data set back to the level of conceptual design methods. The conceptual design module is object-oriented and concepts, both for parameter and method replacement, are introduced. First results for multi-fidelity calculations are shown.

### TABLE OF CONTENTS



## 1. INTRODUCTION

Current design processes in aircraft design do not only spread between disciplines but cover also multi-fidelity and multi-scale aspects. This three-dimensional structure is introduced by La Rocca [12]. A change in the level of fidelity may also cover the generation of sufficient data to trigger further analysis modules and hence increase the bandwidth of disciplines covered.

A combination of analysis modules does not necessary lead to a coherent description of a product or converging solution of an analysis chain. Gaps may remain in the named multi-dimensional design processes. A calculation of component masses, for example, is dependent on overall masses. The overall masses however can not be computed by analysis modules for component masses. Closing the gaps and offering synthesis capabilities for the design led to the demand for a modular conceptual design code, as changes in the design process may occur frequently.

Three different fidelity levels in airplane design are distinguished in the present study. Conceptual design so far mostly consists of history-based empirics, increasing the design space from a set of requirements to a first concept for a new design. The number of parameters is generally small and analogous models are applied. Methods used in conceptual design are also known as handbook methods. The models used in preliminary design are usually lower order physics-based models. The application of these models mostly requires a geometry definition as input, while runtimes and the amount of data are still sparse. Finally, in this work detailed design is seen as the phase where high level models are used to run the analysis.

As stated by the AIAA Design Engineering Technical Committee in 2004 [23] the current phase of aircraft design is driven by physical models. Multi-disciplinary techniques, mostly based on physical models, have been under ongoing development in the aircraft design community. These design environments are supposed to deliver more information at early stages of the design cycle, not taking into account that the computation time is still significant.

Nevertheless, the initial design for a new aircraft configuration is still carried out using conceptual design methods. La Rocca [12] calls this the diverging phase of design due to

*Corresponding Author: daniel.boehnke@dlr.de

the fact that the design space is enlarged to take into account various new concepts and technologies. Conceptual design codes are based mostly on handbook methods that are derived from existing aircraft. It can be estimated that the design space described by the handbook methods is limited to the underlying database. Additionally, the target function for the optimization and design of aircraft in the database is not known, e.g. limitations may occur from family design and production. The available design space for new technologies and concepts is therefore limited. Where physical correlations are applied, these may not be sufficient to reflect the described technology.

Depending on the required amount of input data and computing time, higher order physical models may be used in preliminary and detail design. Several studies have been carried out to link higher order methods with fast geometry generation tools. These efforts have enabled the automatic use of physical methods, as most of the work-intensive reoccurring preprocessing is automated. Examples can be seen in knowledge-based engineering and rule-based systems. It can be expected that computational intricate and expensive tools are used in a later phase of the design, described by La Rocca as the converging phase.

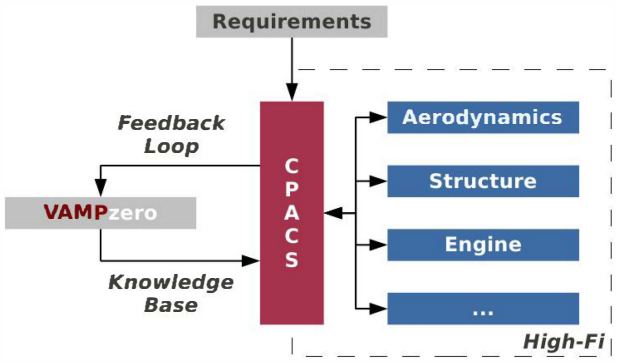


**Figure 1**. Approach to Multi-fidelity

The goal of this study is to show the development steps necessary to close a multi-fidelity design loop, as shown in Figure 1. Part of this is the software structure within a conceptual design module, as well as the surrounding distributed design environment. Communication within the environment is established via a central model. These parts of the distributed design environment will be explained further in the following sections. The first application demonstrates the closed loop from the top-level requirements to the data set that is sufficient for further analyses. Results for a stand alone version against a combined multi-fidelity solution will be shown for the comparison of the design loop. In this way, a prototype multi-fidelity design loop is established in the current design environment at DLR.

This paper is organized as follows: Section 2 gives a brief overview over available design methods in conceptual aircraft design and design frameworks known in the industry and at research institutions; Section 3 gives the design framework of DLR introduced together with the central model for aircraft design; the conceptual design module developed is elaborated on Section 4 and processes for up- and downscaling, which are the core of this paper, are described in this section as well. The results and discussion are presented in Section 5. Finally, an outlook on future activities is given in Section 6.

## 2. State of the Art

So-called handbook methods have grown up to standards in aircraft design. Analysis modules based on these methods are used due to their ability to generate results in a fast manner. Additionally, these tools can run on the base of a high level parameter set, respectively the top level aircraft requirements (TLAR). The input of mission requirements with basic geometric information is sufficient for the first calculations. Based on some elementary equations, e.g. Breguet-Range, formalisms have been built up that can be found in the well-known literature originating from Raymer [17], Roskam [20] and Jenkinson [8]. As can be seen from up-to-date publications, e.g. Filippone [7] and Kundu [11], research in the field of preliminary and conceptual aircraft design methods is an ongoing topic. Most of the standard handbook methods come with an additional software package. The same is true for RDS by Raymer and Advanced Aircraft Analysis (AAA) by Roskam. Additionally, the Flight Opimization System (FLOPS) by McCullers [16] is a conceptual aircraft design code to be named. Kroo, *et al* [10] developed the Program for Aircraft Synthesis Studies (PASS). PASS integrates advanced architecture and approaches to knowledge-based engineering. Other commercial solutions to conceptual aircraft design include Piano[3] and J2Aircraft[4] as well as Pace[5]. Some of the listed codes feature a graphical user interface while others are interfaced via namespace files or commandline prompts. The formalisms used in these codes can be described as semi empirical, as some of the equations may be physics-based but still constrained by values taken from statistical data, e.g. thrust-specific fuel consumption.

Although many valuable design tools exist in the various disciplines, it turns out that bringing them together is a difficult and demanding process. Design environments need to define a common namespace and design codes need to adopt this to their own code. Model driven architectures are known from software engineering and Reichwein, *et al* [18] link this expression to engineering design. The elementary idea remains unchanged, as all design models need to be deduced from the central model (common namespace). Multidisciplinary frameworks to be named include the Design Engineering Engine (DEE) from TU Delft [4], [24] as well as the Multidisciplinary Design Optimization System (MDOPT) from Boeing by Le Doux [14]. The already mentioned PASS module is coupled with modules for environmental analyses in the Collaborative Application Framework for Engineering (CAFFE)

[3] www.lissys.demon.co.uk
[4] www.j2aircraft.com
[5] www.pacelab.de

as shown by Antoine, *et al* [2]. In their study Alonso, *et al* [1] present a framework for high-fidelity applications coded in Python. For research in the field of design languages the DesignCompiler 43 is in use at the University of Stuttgart. An application for multi-disciplinary analysis in aircraft design is shown in [6]. At DLR a multi-disciplinary engineering environment based on CPACS is under development. Liersch, *et al* [15] have presented first results. Applications in the area of environmental research in respect to the air transportation system are shown by Koch, *et al* [9].

Several definitions of the term knowledge based engineering (KBE) can be found in the literature. The most ongoing is given by La Rocca, *et al* [13] stating that: *KBE can be defined as a technology that allows capturing product and process multidisciplinary knowledge by means of integrated software applications that can automate the repetitive design activities, thereby reducing engineering time and cost.* The overall idea is the automation of reoccurring design tasks which hamper the design process and are a source for failure due to repetitive human interaction. Several approaches use automated geometry generation and model buildups for physics-based analysis. KBE is therefore connected to most multi-disciplinary projects as it enhances the model conversions. Applications can be found by Rodriguez, *et al* [19] and Sarakinos, *et al* [21] for airplane geometry generation and also by Sensmeier, *et al* [22] for wing structural sizing.

## 3. Design Environment

The design environment at DLR is subdivided in three components. These cover the framework for process control and optimization, a common namespace (central model) for the exchange of information and the various analysis modules. At DLR the focus is currently on setting up a design environment built on non-proprietary components. At a future point in the development process of the described design environment an open-source distribution is planned.

The integration framework consists of two parts: Firstly, the editor and visual environment for the creation, modification and access control of analysis tool chains. This graphical user interface provides some kind of workspace and enables process designers to interact with analysis modules. This encompasses coupling of modules as well as interactions with central model representations, regardless of whether textual, structural or geometric. Secondly, there is the core logic that provides data transfer between remote components, management of intermediate and resulting data sets, extraction and merging of partial data with the central data model. Further duties that the framework supports are convergence control and optimization.

Expertise and, therefore, analysis capabilities are spread across the boundaries of institutions and locations. For this reason it is necessary to provide resources so that analysis modules can be triggered remotely, e.g. interfacing an analysis module for aerodynamics at DLR in Braunschweig from a desktop in Hamburg. Design teams in distributed institutions benefit, as not only the product description is standardized but the coupling of analysis modules is also identical. Process information, as often coded in from of analysis chains, can be exchanged. Currently, a proprietary framework (ModelCenter by Phoenix Integration)[6] is in use at DLR, but the Reconfigurable Computing Environment (RCE)[7] is under development and is supposed to replace the current solution by an in-house open-source approach.

Due to the fact that in efficient data exchange the number of interfaces is the critical factor for the flexibility of a design environment, a central information model is a key feature. The structure of the coupling of analysis modules has a significant effect on the number of interfaces as shown in Figure 2. Not only does the number of interfaces increase but the analysis modules also get more independent. Hence changes in one module do not necessarily have impact on other tools. The central information model reflects the common namespace of the design team and can be seen as the meta-model for all of the deduced analysis models.

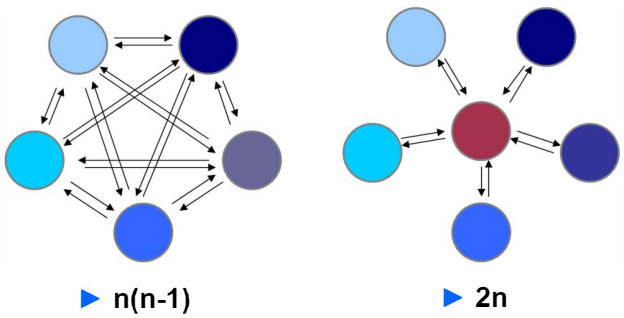


**Figure 2**. Number of Interfaces in Distributed Design

A central model for a design environment as described above consists of two aspects. On the one hand, the elements, attributes and their structure need to be defined in a schema definition. On the other hand, the explicit content has to be stored in a data set which conforms to the schema definition. Whereas the data set is mainly used for the exchange of information, the schema definition is utilized for documentation, model validation and model generation.

DLR has been developing the Common Parametric Aircraft Configuration Schema (CPACS) since 2005. It holds detailed constructs for the exchange of information on the level of preliminary design. The numerical system is in operational use at all aeronautical institutes of DLR and has been extended for civil and military aircrafts, rotorcrafts, jet engines and entire air transportations systems. An application of the design environment is shown by Liersch, *et al* [15]. CPACS is based on XML technologies, an assessment of alternatives for modeling languages for preliminary aircraft design is shown in [5].

[6] www.phoenix-int.com/software/phx_modelcenter.php
[7] www.rcenvironment.de

The range of entities modelled in CPACS goes beyond the representation of base datatypes. It holds detailed interpretable type definitions for semantic entities from the air transportation system. Precise definitions for geometric elements like aircraft, wings, sections, elements, profiles, points, and transformations are given. These entities cover the bandwidth of the named projects. Additional elements defined include amongst others airports, airline fleets and flightplans. A documentation is available. However, working with a dataset of this size and complexity could be simplified by providing an automatic mechanism known from knowledge-based engineering to create CPACS data sets.

CPACS not only holds information concerning the considered object but also data for the connected analysis module. This *tool-specific* data is transferred along with the dataset and carries further information to trigger different options in the analysis modules. In this way, analysis modules can interact with each other in a sequential way. Additionally, the framework allows for splitting and merging CPACS datasets. *Tool-specific*, as well as other datasets, can therefore be added before or after an analysis module run.

Within the present study analysis modules for aerodynamics, engine and conceptual design will be used. From the aerodynamic calculations, an aerodynamic performance map varying in Mach- and Reynolds number is retrieved providing information for force and moment coefficients for different attack and yaw angles. The engine analysis module delivers among global engine parameters a performance map for the thrust-specific fuel consumption in dependence of flight altitude, mach number and thrust. The conceptual design code is described in detail in Section 4. These modules are coupled distinguishing between higher and lower level methods, with the latter being on the low level part, as displayed in Figure 1. In this case, higher level modules are identified by the fact that they take their input natively from the detailed granularity of CPACS.

The design loop is initialized by a TLAR dataset triggering the conceptual design module (VAMPzero). For the upward change in level of detail a knowledge-based approach is used for the generation of CPACS files with the use of VAMPzero. This includes the geometry generation, as well as additional data such as the mass breakdown and the tool-specific inputs for further analyses in more complex modules. The feedback loop is closed downwards by reducing the granularity from the CPACS data set back to the level of conceptual design methods. In this way the results from the analysis modules are fed back into VAMPzero for the synthesis of the design.

## 4. Conceptual Design Module

The request for a new conceptual design tool originates from DLR's multidisciplinary aircraft design processes. Although several higher order models are already coupled with CPACS, as described by Liersch, *et al* [15], some conceptual methods are required to close the gaps, e.g. deriving center of gravity locations from the input of component-based analysis modules, in the holistic design process. Consequentially, the new code needs to feature requirements such as batch capabilities and should be modular, reusable, readable and extensible.

In this section, the module structure of the developed conceptual design code is elaborated on. The structure is elementary for interactions of VAMPzero with other analysis modules inside the design environment. The methods for up- and downscaling are described subsequently. These are an irreplaceable part when it comes to closing a multi-fidelity design loop. In this paper, strategies and methods are described. First results are shown in Section 5 and further examinations will be part of future research.

*Module Structure*

An object-oriented structure was chosen for the development of the new conceptual design code (VAMPzero) to increase the modularity and to be able to detach the feature aspects (file handling, convergence control, process control) from the design aspects (parameter definitions, calculation methods). In this way a code is introduced that can easily be extended and adapted.

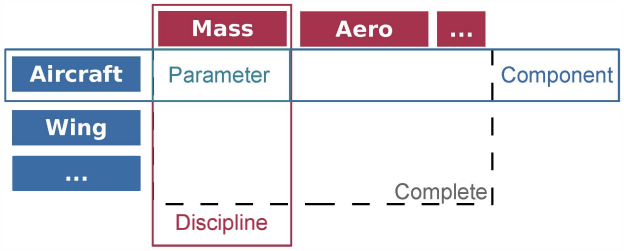


**Figure 3**. VAMPzero Object Oriented Structure

The two main classes implemented in VAMPzero are `components` and `parameters`. These two classes hold generic routines that are inherited or overwritten by specific classes, e.g. a wing `component` class or a span `parameter` class. `Components` share associations with other `components` and are structured in a hierarchical way. Each `component` holds several `parameters` that are grouped by `disciplines`. A wing that is modeled as a `component` has a `parameter` span within the `discipline` geometry. The structure is displayed in Figure 3. Each of the named elements holds a calc method. The calc methods in `components` and `disciplines` are used for process control, whereas the `parameter` calc methods hold the design knowledge mostly in empiric or analytic form. In this way it is possible to use parts of the analysis capabilities in a transverse way and replace `components`, `parameters` and `disciplines`. This is an essential part of the aspired multi-fidelity strategy and will be described in the following sections in more detail.

In a baseline version of VAMPzero, which can be run stand alone, the `parameter` calculation methods base on handbook methods similar or identical to some of the conceptual

design codes that have already been mentioned previously, [20], [17], [3], [11]. At some points (e.g., mass estimations) new correlations are introduced to include databases that fit our design purposes better.

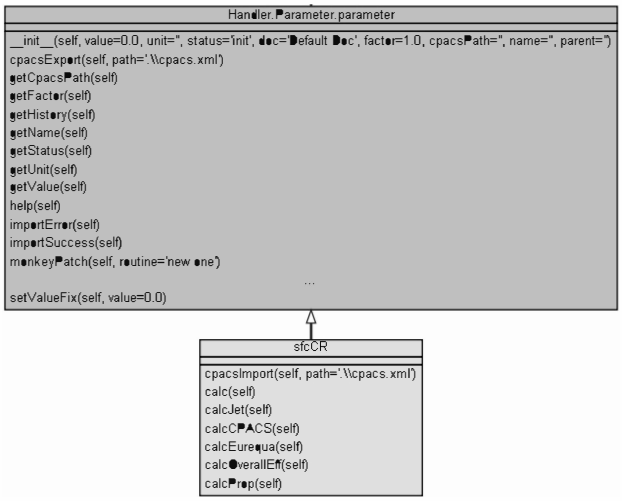


**Figure 4**. Class Diagram: SFC in Cruise

Additionally, several other methods are available for both `components` and `parameters`. These are coded in the respective classes. Depending on the implementation of the subclass an arbitrary number of methods can be overwritten. The calculation process for example can be broken down into: importing a value from CPACS, calculating until converged, exporting to CPACS. As a matter of fact, the calc method needs to be overwritten for each new class. For the import and export methods, either XPath variables can be specified or new methods are implemented. An example is shown in Figure 4. The UML representation is taken from the generated documentation[8]. The `parameter` class holds several generic methods and some of them are overwritten by the class for the thrust-specific fuel consumption, namely the import method and the calc method. A third calcCPACS method is specified as well which will be elaborated at a later point in this paper.

Changing the calculation method of a `parameter` often changes the dependencies in the model. As the calculation method accesses the values of other `parameters`, it is possible to track the adjacencies in the model during runtime. The representation of the adjacencies is made using open-source mind-mapping software[9] and supports the transparency of the analysis module. Links between `parameters` in the model can be displayed without looking into the code. As different calc methods are available within a `parameter` and are chosen depending on the status of other `parameters` and available input from CPACS, this representation reflects the analysis process even better than the underlying code. Examples of this representation can be seen in Figures 5 and 6. The adjacencies are displayed for the `parameter` in the center with just one plane of entries. To the left hand side `parameters` accessing the value of the central `parameter` (caller[10]) are aligned. For the calc method of the `parameter`, the `parameters` on the right hand side are accessed (callee). The views can be navigated in a browser-like way and help in understanding the recursive structure of the design process.

Knowing the dependencies in the model is the basis for more operations to monitor the calculation. When an analysis run is set up, `parameters` are assigned a status that can either be `initialized`, `calculation` or `fix`. At the construction of the objects, each status is set to `initialized`. If a `parameter` is not fixed its value will be calculated by one of its calc methods. Fixing `parameters` in the model may lead to inconsistencies. These occur whenever all inputs of a parameter with status `fix` are fixed as well. The takeoff mass, for example, is derived from the payload mass, the operational empty mass and the mission fuel mass. If the latter three values are fixed, the takeoff mass can be calculated. If the takeoff mass is `fix` as well, a warning will be given.

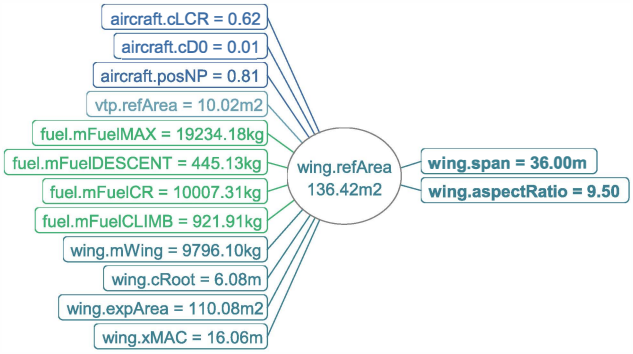


**Figure 5**. Dependencies for geometric definition

Depending on the status of the model, different calc methods are chosen. If the `parameters` span and aspect ratio of a `component` wing are fixed (by user input), the reference area of the wing is calculated by geometric definitions. An example for this calculation can be found in Figure 5. On the right hand side the fixed (bold font) `parameters` span and aspect ratio are displayed. The left hand side shows all `parameters` derived from the reference area.

In a different calculation run, the span may not be fixed as it is supposed to be calculated by VAMPzero. In this case a different calculation method for the reference area is chosen depending on the maximum wing loading and the maximum takeoff mass. The results for the calculation run are displayed in Figure 6. In this setup the status of right hand values is also `calc`. The list of parameters on the left hand side is extended by the span as it is now determined from the reference area.

[8] epydoc.sourceforge.net
[9] sourceforge.freemind.net

[10] Caller and Callee are common terms in code profiling. In VAMPzero only `components` and `disciplines` may call calc methods. `Parameters` may only fetch their values but not trigger calc methods of other objects.

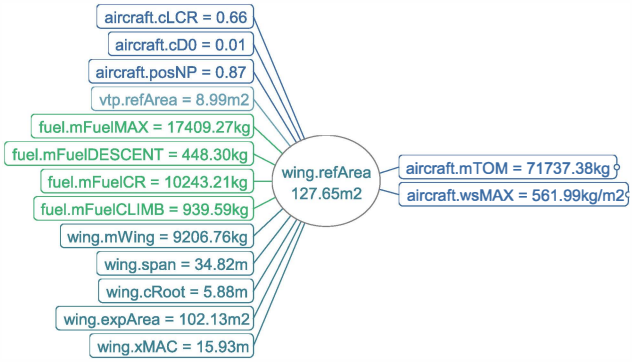


**Figure 6**. Dependencies for sizing

When developing conceptual design codes or multi-disciplinary design processes usually just a few key values, e.g. maximum takeoff mass, operation empty weight and static thrust, are checked for convergence. In VAMPzero convergence is checked for all `parameters` separately, as can be seen in Figure 7. The relative change of the value of a parameter is displayed dependent on the number of iterations. All graphs[11] are displayed on a logarithmic scale to improve the clarity of the diagram. The displayed convergence histories are shown for an analysis run with a minimum number of input values. As a matter of fact, this calculation requires several iterations but shows the numerical stability of the code.

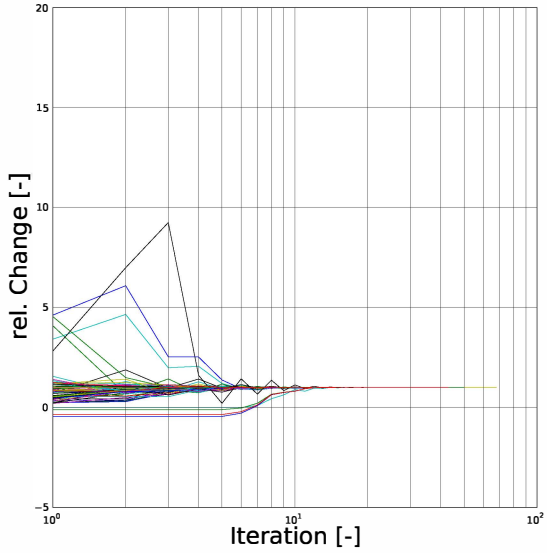


**Figure 7**. Convergence History for Parameters

*Knowledge Base*

As already outlined, one of the requirements for a conceptual design module is closing the gap between TLAR and a dataset sufficient enough to trigger higher level methods. An increase in information is equivalent to a transfer from a system with lower dimensionality to a system with more dimensions. Hence this can only be carried out using additional (design) knowledge. By utilizing techniques from KBE this process can be coded and automated.

For the purposes of this project it was important to generate the parametric geometry description as defined by CPACS out of the reduced parameter set of the conceptual design module. As we are currently focussing on civil jet transport, a sufficient database to retrieve rules for a KBE approach is available. Additionally, data generated by VAMPzero, mainly mass distributions, moments of intertia, as well as *tool-specific* data are transfered to CPACS. Thereby product and process knowledge are used.

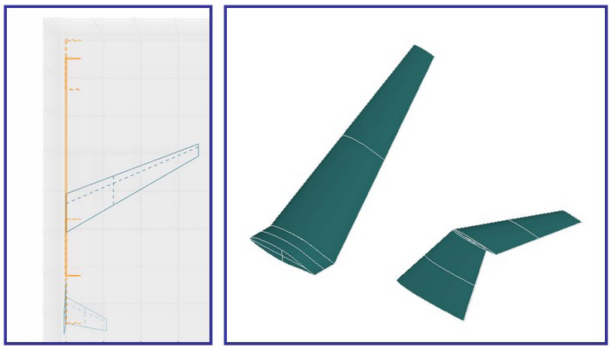

**Figure 8**. Geometry in VAMPzero (left) and CPACS (right)

An example for a cpacsExport method that was implemented to the wing component can be seen in Figure 8. As the horizontal and vertical tail inherit the wing component, the same methods can be applied easily. The upward change in level of detail can only be realized by applying knowledge-based rules and hence this process should usually be monitored by an *engineer-in-the-loop*.

Both CPACS and VAMPzero are coded in object-oriented languages. Although CPACS is more a schema definition than tool it was possible to extract libraries from this definition that allowed for easy handling of objects. Generating CPACS-like objects and exporting them in the correct syntax is therefore integrated in VAMPzero without implementing additional code fragments.

*Feedback Loop*

While the previous subsection elaborated on a technique to derive a description of higher granularity out of a conceptual design module, this subsection focuses on reusing the information generated by higher level tools. This data needs to be reassembled in order to be able to perform a synthesis of the design within the analysis process.

Two different approaches will be shown: the first being the replacement of the value for a single parameter by importing (or deriving) this value from CPACS. And the second approach where replacing the value of the parameter is not sufficient. If

[11] In the current version more than 250 `parameters` are declared. The legend is therefore excluded in Figure 7.

additional dependencies of the higher level information need to be resolved, a new calculation method needs to be introduced. In this case the calculation method is replaced during runtime of the code.

*Parameter replacement*—The influence of the Oswald factor on the global characteristics of an aircraft design is ineluctable. In Equation 1 taken from Raymer [17] the Oswald factor ($e$) is determined from the aspect ratio ($\Lambda$) and the leading edge sweep angle ($\varphi$). In this case the calculation method is limited to leading edge sweep angles larger than 30°.

$$e = 4.61(1 - 0.045\Lambda^{0.68})(cos(\varphi))^{0.15} - 3.1 \qquad (1)$$

$$with \quad \varphi > 30°$$

For calculations in a design space that can not be accessed by this equation, a different method must be used. Physical models are preferred as limitations due to geometric parameters are avoided. Due to the possibility to export geometric information to CPACS as seen in the previous subsection, aerodynamic analysis modules can be used to derive performance maps. In the case of fast linear methods it is easy to derive the Oswald factor for the configuration. In this case more data is imported from CPACS as can be reflected by parameters and methods in VAMPzero. Contrary to an upward change in the dimensionality as in KBE, this operation is well-defined.

Less complex imports are possible if no change in the dimensionality occurs and parameters are modelled non-ambiguously in the central model and the conceptual design module, e.g. for the design cruise Mach number. If the `parameter` is imported, its status is set to `fix` neglecting further calculations within VAMPzero. In this case the analysis capability is within the design environment and not available within the conceptual design code. If parameter variations occur, the analysis process needs to be triggered again to update the imported value.

*Method replacement*—Depending on the state of a model different calculation approaches need to be made for single `parameters`. For example, the aspect ratio can either be estimated from sizing requirements or calculated exactly if span and reference area are given in a model. As VAMPzero is capable of exchanging its calc methods during runtime, this capability can also be utilized for multi-fidelity aspects.

In object-oriented modelling, entities are described by classes uniting both the attributes and methods that describe an entity. Attributes hereby describe the state of the class, respectively the object at runtime. The behavior is represented by the implemented methods. As already mentioned, each class in VAMPzero holds a calc method. To adapt to the current state of the model the calc method of a `parameter` can be replaced by other methods. This procedure of exchanging a reference to an executable block of code without the owning object knowing is renowned as *monkey-patching*[12]. Dynamic object-oriented programming languages like Python, Ruby or JavaScript support *monkey-patching* with their ability to handle functions (or references to executable blocks of code, known as closures) as objects. They are therefore able to replace them at runtime, which can be considered some kind of self-modification, since the method replacement was never written down in the source code. VAMPzero is coded in Python where all entities are declared as functions. Hence the *monkey-patch* can be coded easily, allowing the module to choose between various coded calc methods.

In Equation 2 a correlation for the thrust-specific fuel consumption in cruise condition depending on the bypass ratio is given by Raymer [17]. The equation is valid for bypass ratios of up to 6.0. This limits the analysis to engine concepts that are not up-to-date. Furthermore, the thrust-specific fuel consumption is an important parameter for the design and hence a calculation method with dependencies in more parameters is desirable. This may help in evaluating influences from different technologies, as their impact is reflected in the respective parameters.

$$SFC_{cr} = 0.88e^{-0.05BPR} \qquad (2)$$

$$with \quad BPR < 6.0$$

As mentioned in the previous section, it is possible to monitor dependencies in VAMPzero at runtime. An example for Equation 2 is given in Figure 9. Direct links exist between the mission fuel mass and the design range in the displayed configuration of VAMPzero. The current calc methods of these `parameters` rely on the the input from the thrust-specific fuel consumption. The only parameter on the right-hand side is the bypass ratio. In the described calc method there are no other influences on the SFC.


**Figure 9**. Initial SFC Dependencies

Fetching values from CPACS is done by cpacsImport methods in VAMPzero. In contrast to the parameter replacement described previously, if multi-dimensional links for a parameter exist in CPACS these need to be imported as well. Hence the calc method needs to be adjusted and the `parameter`'s status will be set to `calc`. The thrust-specific fuel consumption is given depending on the Mach number, flight altitude and required thrust in CPACS. The import method reads these values and stores them inside the `parameter`. Afterwards, the cpascImport method exchanges the calc method as shown

[12]Calc methods in VAMPzero do not hold on any input `parameters`. Associations to other `parameters` are created within a method. The *monkey-patch* is therefore not to be mingled with an overloading of methods as can be done e.g. in Java. If the calc method of a `parameter` in VAMPzero changes (or is exchanged), no adjustments need to be made in other parts of the code

in Equation 2 by calcCPACS that interpolates in the stored performance map. Figure 10 shows the new dependencies inside the model.

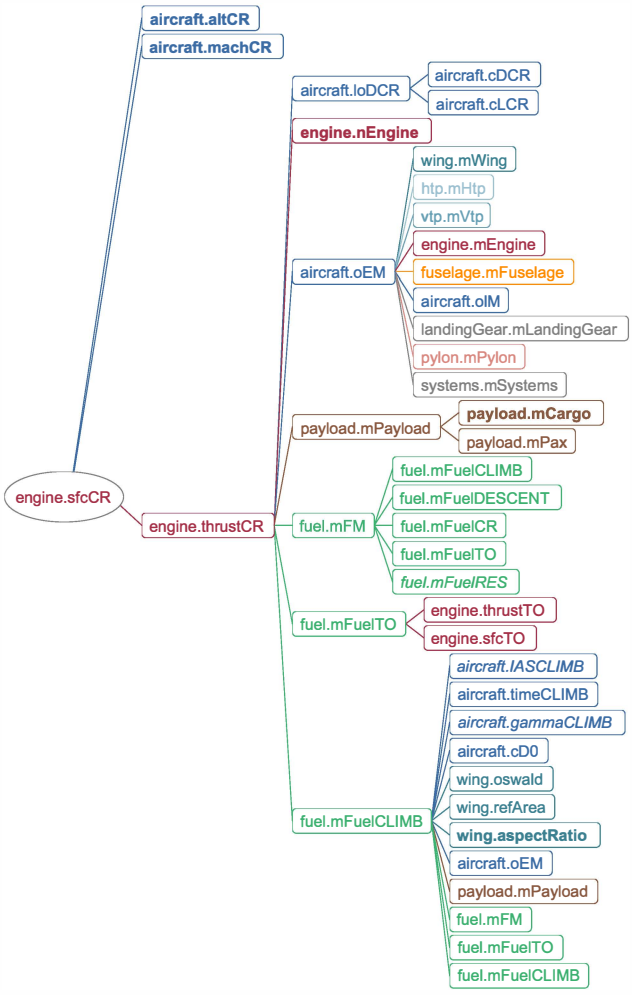


**Figure 10**. SFC Dependencies after Monkey Patch

## 5. Discussion

In the context of this paper the current design environment at DLR is introduced, focusing on the central model CPACS and a conceptual design module VAMPzero.

Benefits from introducing object oriented modeling into a conceptual design module include the flexible structure of the code as well as an increased transparency. Working in object oriented environments demands adjustments from developers, especially as engineers are more familiar with procedural calculation. Nevertheless, it has been experienced that people working with the code could start implementing new parts only after a short training. Considerable advantages come from the fact that the separate code fragments in each file are very short and the interdependencies are easily traceable either via the introduced visualization or simply by the advanced editing[13,14] capabilities.

[13] www.eclipse.org
[14] pydev.sourceforge.net

Capabilities of VAMPzero have been extended to enable the generation of CPACS files from a reduced set of parameters making use of KBE techniques. Due to the fact that the generated geometry is not a result of an upscaled simpler description but of the sizing logic coded in form of the handbook methods, it is likely that first estimates are within the range of converged designs.

It is obvious that introducing more dependencies in a model will lead to results more sensitive to the inputs. This can be seen in Figure 11 displaying results in the model depending on the aspect ratio ($\Lambda$) and cruise altitude. Graphs in the figure display values for the thrust-specific fuel consumption (continuous) and the takeoff mass (dashed) both for the level 1 (multi-fidelity, import from CPACS) and level 0 (low-fidelity, handbook) calculation methods introduced in Section 4. It can be seen that for the thrust-specific fuel consumption, constant values result from the level 0 equation whereas at level 1 a direct link to the cruise altitude exists and changes in the aspect ratio are linked via the required thrust. For the takeoff mass an offset occurs at level 0 in contrast to, again, a direct link in the level 1 calculation. Here not only an offset is present due to the change in aspect ratio but also more dependencies from the cruise altitude.

Available handbook methods as shown in Equations 1 and 2 can be replaced by more advanced methods that can be found in more specific literature and this is already done in some cases within VAMPzero. Nevertheless, it is requested not to let the conceptual design code grow up to yet another monolithic and, hence, inflexible analysis module.

## 6. Outlook

First applications for the knowledge-based approach were shown for wing geometries and *tool-specific* data. Future work will enlarge the scope of the knowledge-based approach to include further geometries (fuselage, engine) as well as further disciplines (structure, performance).

Additionally, the number of import methods will be extended to include further analysis modules. Only in this way can it be assessed whether the taken approach for VAMPzero proves to be as modular as desired. In a highly loaded scenario only some elementary equations will remain and the question arises if a conceptual design module or rather a simple script are reasonable choices for the synthesis of the design.

As soon as the approaches shown in this paper have been extended in the described way, a complete multi-fidelity design of new configurations can be aspired. The goal should be to model techniques with respect to a new configuration that cannot be assessed reliably by handbook methods to force the use of physical models. Possible projects include forward swept wings and new concepts for rear mounted engines.

To promote an integrated solution to a multi-fidelity, multi-disciplinary aircraft design environment a coupling of the

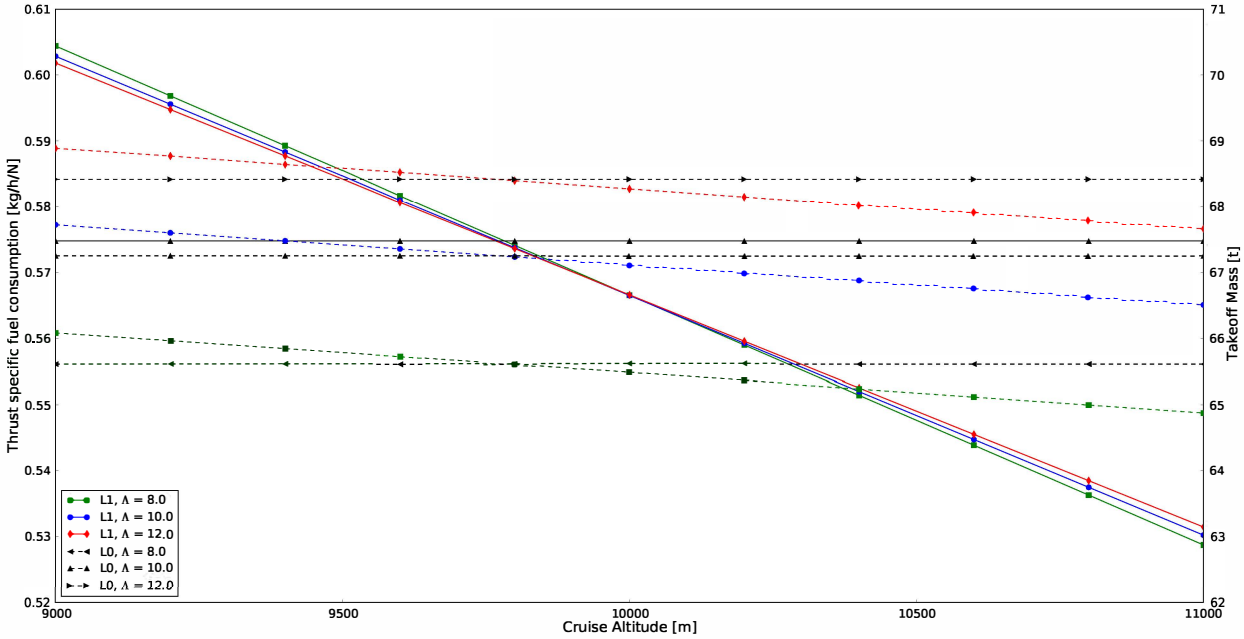


**Figure 11**. SFC and TOM for Low (L0) and Multi-Fidelity (L1)

framework, VAMPzero and CPACS is aspired. In this scenario a graphical user interface integrated into the framework will enable a user to access parameters within VAMPzero. Once VAMPzero is configured in this way it can be used to generate a valid CPACS file that can then be coupled with other modules integrated into the design environment. The goal is to simplify the work with CPACS and enable the user to derive and analyse first designs in a quick way. Furthermore, this package is supposed to be used as a basis for cooperation with other research institutions and to be distributed as open-source. Contributions from outside of DLR may assist in setting up a parametric holistic description of the air transportation system in CPACS and exchange the abilities of different analysis modules.

## REFERENCES


[1] ALONSO, J., LEGRESLEY, P., VAN DER WEIDE, E., MARTINS, J., AND REUTHER, J. pyMDO: A Framework for High-Fidelity Multi-Disciplinary Optimisation. *10th AIAA / ISSMO Multidisciplinary Analysis and Optimization Conference* (2004).

[2] ANTOINE, N., KROO, I., WILCOX, K., AND BARTER, G. A Framework for Aircraft Conceptual Design and Environmental Performance Studies. *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* (2004).

[3] BALLUFF, S., PFAFF, J., RUDOLPH, S., AND VOIT-NITSCHMANN, R. Ermittelung der Leitwerksflaechen im Flugzeugvorentwurf unter Einsatz des fallbasierten Schliessens. *DGLR Congress* (2009).

[4] BERENDS, J., AND VAN TOOREN, M. Design of a Multi-Agent Task Environment Framework to Support Multidisciplinary Design and Optimisation. *ASM* (2007).

[5] BOEHNKE, D., LITZ, M., NAGEL, B., AND RUDOLPH, S. Evaluation of Modeling Languages for Preliminary Aircraft Design in Multidiciplinary Engineering Environments. *DGLR Congress* (2010).

[6] BOEHNKE, D., REICHWEIN, A., AND RUDOLPH, S. Design Language for Airplane Geometries using the Unified Modeling Language. *ASME Int. Design Engineering Technical Conferences (IDETC) & Computers and Information in Engineering Conference (CIE)* (2009).

[7] FILIPPONE, A. Theroretical Framework for the Simulation of Transport Aircraft Flight. *Journal of Aircraft 47*, 5 (Sept. 2010), 1679–1696.

[8] JENKINSON, L., SIMKIN, P., AND RHODES, D. Civil Jet Aircraft Design. *Butterworth Heinemann* (1999).

[9] KOCH, A., NAGEL, B., GOLLNICK, V., DAHLMANN, K., GREWE, V., KÄRCHER, B., AND SCHUMANN, U. Integrated analysis and design environment for a climate compatible air transport system. *AIAA Aviation Technology, Integration and Operations Conference* (2009).

[10] KROO, I., AND TAKAI, M. A Quasi-Procedural, Knowledge-Based System for Aircraft Design. *AIAA 6502* (1988).

[11] KUNDU, A. K. Aircraft Design. *Cambridge Aerospace Series* (2010).

[12] LA ROCCA, G., AND VAN TOOREN, M. Enabling distributed multi-disciplinary design of Complex Products : a knowledge based Engineering Approach. *Journal of Design Research 5* (2007), 333–352.

[13] LA ROCCA, G., AND VAN TOOREN, M. Knowledge-based engineering to support aircraft multidisciplinary design and optimization. *Journal of Aerospace Engineering 224* (2010), 1041–1055.

[14] LEDOUX, S., HERLING, W., FATTA, G. J., AND RATCLIFF, R. MDOPT- A Multidisciplinary Design Optimization System Using Higher order Analysis Codes. *AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* (2004).

[15] LIERSCH, C., AND HEPPERLE, M. A Unified Approach for Multidisciplinary Aircraft Design. *CEAS European Air and Space Conference* (2009).

[16] MCCULLERS, L. A. Flight Optimization System (FLOPS) User's Guide. *NASA Langley Research Center* (2009).

[17] RAYMER, D. Aircraft Design: A Conceptual Approach. *AIAA Education Series* (1989).

[18] REICHWEIN, A., AND HERTKORN, P. On a model driven approach to engineering design. *International Conference on Engineering Design* (2007).

[19] RODRIGUEZ, D. L., AND STURDZA, P. A Rapid Geometry Engine for Preliminary Aircraft Design. *44th AIAA Aerospace Sciences Meeting and Exhibit* (2006).

[20] ROSKAM, J. Airplane Design. *DARCorporation I-VII* (1989).

[21] SARAKINOS, S., VALAKOS, I. M., AND NOIKOLOS, I. K. A Software tool for parameterized aircraft design. *Advances in Engineering Software 38* (2007), 39–49.

[22] SENSMEIER, M. D., AND SAMAREH, J. Automatic Aircraft Stuructural Topology Generation for Multidisciplinary Optimization and Weight Estimation. *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference* (2005).

[23] TAM, W. F. Improvement Opportunities for Aerospace Design Process. *AIAA Space Conference & Exhibit* (2004).

[24] VAN DER LAAN, T. Knowledge based engineering support for aircraft component design. *PHD Thesis TU Delft* (2008).

## BIOGRAPHY

**Daniel Böhnke** received his Diploma Degree in Aeronautical Engineering from the University of Stuttgart in 2009 focusing on Aerodynamics and Data- and Knowledge Management. In 2007 he was an Erasmus Fellow at the University of Southampton. He is currently a Ph.-D. student in the Integrated Aircraft Design Group within the Institute for Air Transportation Systems in the German Aerospace Center (DLR) at the Technical University Hamburg. His interests include conceptual design, multi-fidelity and collaborative engineering.

**Björn Nagel** received his Diploma Degree in Aeronautical Engineering from the Technical University of Braunschweig in 2003. Following, he worked as scientist at the German Aerospace Center (DLR), Institute of Composite Structures and Adaptive Systems in Braunschweig. His work was focussed on the design and optimization of passive and active light weight structures being subject to aeroelastic interactions. In 2007 he joined the then founded DLR Institute of Air Transportation Systems in Hamburg as head of the department Integrated Aircraft Design. With his team he investigates aircraft for future scenarios and novel technologies. One major interest is constituted by methods for collaborative research in large frameworks of specialists within DLR and beyond.

**Volker Gollnick** received his Diploma Degree in Mechanical and Aerospace Engineering from the Technical University of Braunschweig in 1991. He started his professional career at the German Forces Flight Test Center as a flight test engineer and research engineer for rotorcraft handling qualities. In 1998 Volker Gollnick became a project manager for engine test rig development at debis systemhouse until he took over the position of head of department at Eurocopter for cockpit systems and simulation. He received his Ph-D.-degree in 2004 from the Technical University of Munich performing a study about the mission and energy demand based comparison of various transportation systems. After some years with Eurocopter Volker Gollnick advanced to a programme manager position and senior expert at EADS corporate research. 2007 he was asked to build up the newly founded entity 'air transportation systems & technology assessment' within the German Aerospace Center at the Technical University Hamburg. His prime interest of research is the mission oriented and functional aircraft design including avionics and the development of new transportation concepts.