USING OPENVSP IN A CONCEPTUAL AIRCRAFT DESIGN ENVIRONMENT IN MATLAB

Sebastian Herbst, Adrian Staudenmaier
Institute of Aircraft Design, Technical University of Munich

Keywords: *ADDAM, ADEBO, Conceptual Aircraft Design, OpenVSP*

Abstract

The increasing variety of suitable solutions in the aircraft design process which are mainly driven by new technologies on component level offer a large number of feasible design options. Concurrently the process complexity increases as well as the demands on a uniform and comprehensible data model for the rising number of potential fixed-wing aircraft configurations.
For that reason the Institute of Aircraft Design of the Technical University of Munich is developing the new aircraft design environment ADEBO (Aircraft Design Box). It is implemented in MATLAB using an object-oriented programmed (OOP) data model for all types of fixed-wing aircraft without any restrictions regarding their geometric shape or overall size and mass. As new part of the design environment OpenVSP is integrated in order to use its visualization and calculation capabilities. By means of a developed interface to MATLAB data can easily be interchanged between the data model and OpenVSP. Consequently the advantages of both tools can be used.

1 Introduction

Today’s aircraft design process has changed significantly, because new technologies require physics-based calculation methods and most semi-empirical methods lack a valid database. Moreover design solutions that might be named unconventional (no tube-wing configuration or fossil fuel burning propulsion system) are desirable and achievable results. For this reason different approaches have been made: Initially it was examined if enhancements of existing design tools like FLOPS [1, pp. 395–412] could also satisfy the demand of handling advanced technologies [2]. Concurrently also new tools have been developed. Data models (e.g. CPACS [3]), workflow management systems (e.g. RCE [4]) or overall design environments (e.g. CDT [5] or SUAVE [6]).

The main purpose of ADEBO is to set up an aircraft design environment which is flexible, extensible and user-friendly for conceptual fixed-wing aircraft design. By using a universal data model and a set of tools from different branches the design environment covers physics-based as well as empirical methods. The individual tool application follows the principle of using a level of fidelity that is as high as necessary and as fast as possible. Neither the design process nor the selection of tools is narrowed by predefined design patterns which guarantees the consideration of novel technologies and leads to future advanced aircraft designs. This increasing variety of process capabilities and design solutions can be handled clearly and is well-structured by the data model ADDAM. General-purpose patterns for tool applications lead to minimized data administration effort and decrease overall process complexity.

As conceptual aircraft design is an engineering discipline which is strongly driven by mission requirements and initial layout estimations, the resulting performance during trade-off studies needs to be aligned with the geometric shape of the underlying concept. Hence the ability to ensure consistency between the visualized aircraft geometry and the used technical data within the calculations is very important. NASA developed a software called OpenVSP [7] to rapidly create geometric aircraft models without spending much expertise and time in traditional Computer Aided Design (CAD) tools. Today OpenVSP is freely available as open source software which makes it suitable for an integration into ADEBO. Besides using the visualization and calculation methods of OpenVSP the export function for *.stl and *.stp CAD files can be exploited in order to generate geometry input files for simulations (e.g. in OpenSceneGraph).

The requirements on using OpenVSP can be divided into hard and soft points and are summarized in Tab. 1 below.

Table 1. OpenVSP requirements

Hard points:	
Automatic communication	MATLAB and OpenVSP shall exchange data through automatic communication.
Consistency	The geometry parameters shall be consistent in both ADDAM and OpenVSP databases.
Expand functionality	ADEBO shall be able to utilize the openVSP export and analysis functionalities.
Soft points:	
Short processing time	The processing time to complete data communication should be minimized.

2 ADEBO – A view inside

ADEBO is an environment for conceptual aircraft design implemented in MATLAB version R2014b©. Due to the utilization of this widespread technical computing language [8] a large target group ranging from students in academia to scientists and engineers in industry can be attained. Furthermore no additional knowledge of other computing or markup languages like HTML, C++ or Python is required. ADEBO can easily be enhanced and extended by standardized patterns, which simplifies rapid developments and adjustments according to new aircraft technologies. The design environment consists of three key elements depicted in Fig. 1. It consists of a unique object-oriented data model [9] in order to take over and minimize data administration effort as well as a set of several tools of different fidelity and purpose. Moreover ADEBO comprises a library containing databases and frequently used calculation functions and methods.

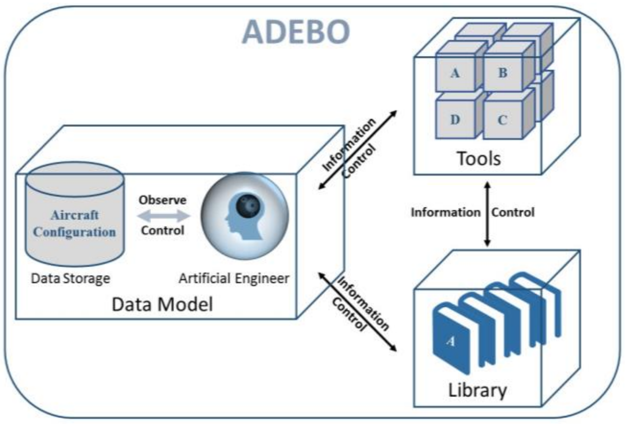

Figure 1. Key elements of ADEBO

The interaction concept of the key elements themselves and a brief description of the working procedure with ADEBO is presented in the next section.

2.1 Concept

In spite of the high flexibility because of using different tools in a design process, a universally valid procedure in working with ADEBO is provided. Initially a data model is required which comprises a configuration object as well as an Artificial Engineer (AE) object (see 2.2). The user only communicates with the aircraft configuration itself or with the AE as the only constant elements of a design process. A unique exception is providing input data directly if an ongoing tool queries additional data. This means set and get data of the aircraft configuration can be carried out directly just as using accompanying operations. For running a particular tool the user commands the AE to start the tool with defined inputs, e.g. configuration object, visualization flag or tool specific input data (for more information see 2.3). As visualized in Fig. 2 the user can easily define a purpose based schedule in order to set up a design process and fill the configuration object with technical data [10].

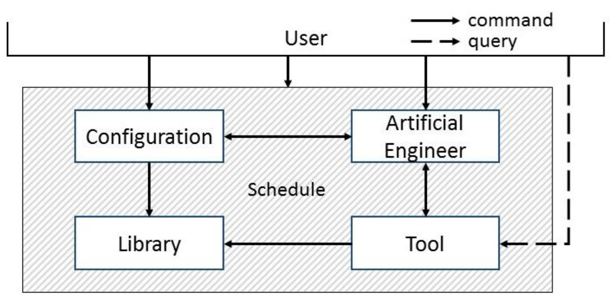

Figure 2. ADEBO command concept

2.2 Data model ADDAM

ADDAM is an OOP data model specifically developed for the conceptual aircraft design process. It consists of two elements: the aircraft configuration and the AE. The configuration object contains all data of the aircraft and fulfills the software requirements of a data model (definiteness, consistency, intelligibility, expandability, compatibility, and transferability) as well as the demands for using specific aerospace nomenclature (follow the "engineer way of thinking", maintain user-friendliness, and guarantee a short training period). In addition the AE contains the complete tool knowledge which links the configuration object with the tools. Furthermore the AE is a kind of observer of the configuration object fulfilling data administration tasks. One major advantage of ADDAM is that every value is assigned to a specific unit which can be of the type SI (standard) or imperial unit system. At any time the unit can be converted or translated by the data model itself. Furthermore each value has a history which records every change in the value itself and its unit as well as its value type and its used calculation method and applied tool in order to determine the value [9, 11].

2.3 Tool utilization

Each tool shall be implemented as efficient and as fast as possible independent from ADEBO. This also means no existing tool needs to be changed or adjusted in order to be suitable for ADEBO. The design environment provides a procedure for the tool integration following the principle of using a specifically developed wrapper for each tool. Figure 3 shows the concept of applying a wrapper for the input and output each. This means the AE calls the tool specific script which automatically runs the input wrapper and starts the tool application.

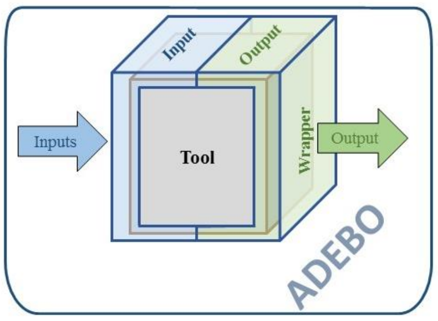

Figure 3. ADEBO wrapper concept

Obviously this wrapper concept generates an overhead in data management and slows down the computational speed compared to running the tool independent from ADEBO. But due to the benefit of being able to use the tool for any kind of aircraft configuration and at any stage of design process without any changes the ADEBO wrapper concept decreases effort, time and also money for carrying out trade-off studies considering multiple aircraft configurations and technologies.

As already mentioned, OpenVSP as standalone software shall be integrated into ADEBO. The main objective is to communicate automatically with the design environment by exporting data from ADDAM into OpenVSP and also import data from OpenVSP into the data model (see Fig. 4)

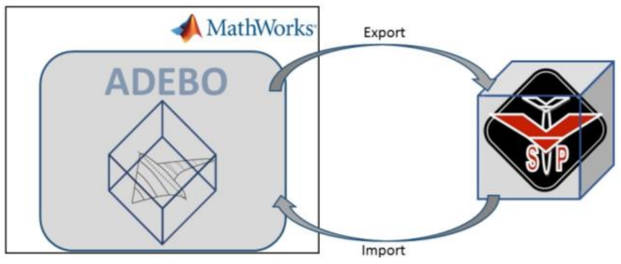

Figure 4. OpenVSP utilization in ADEBO

3 Integrate OpenVSP into ADEBO

Integrating OpenVSP will increase the capabilities of ADEBO by establishing *.stl files easily, using its geometric and aerodynamic calculation functions as well as the ability to create mesh models for initial computational fluid dynamics (CFD) calculations. OpenVSP has many capable and viable options to communicate with MATLAB concerning the integration into ADEBO. The implementation of the communication is highly dependent on the requirements that are derived from the design environment and on the capabilities of OpenVSP. The main objective is to achieve automated communication between MATLAB and OpenVSP which allows passing geometry data between both, compare Fig. 4. Obtaining a connection requires extensive research in the application interface (API) documents [12–17]. Table 2 and 3 show an excerpt of techniques from the API which can be used within MATLAB and other programmable software like Microsoft Visual Studio. Although there are more techniques than displayed below, these are the most beneficial ones in terms of feasibility.

Table 2. Techniques accessing OpenVSP API

Technique	Explanation
OpenVSP user interface	The user interface requires manual inputs resulting in limited automation. However the user interface has the most capabilities since not all "buttons" and commands are existent or documented in the scripting language.
Command line batch mode	Batch mode is basically OpenVSP without the user interface. The command line can process design files, *.vsp3 files and *.vspscript files that commend OpenVSP behavior.
Design file	A design file can be opened in a normal text editor. The MATLAB commands *read/write* and *fopen* can be used to create it. The file can either be created in the GUI or in a *.vspscript file which results in a *.des file. Design files consist of handpicked parameters and their values as well as a specific identifier [13].

Table 3. Techniques accessing OpenVSP API

Technique	Explanation
Scripting	This is the backbone of the OpenVSP API. C/C++ syntax is applied and it is called AngelScript [18]. The documentation contains all available functions that a script can run [14]. Using a script offers the possibility to create, manipulate and calculate geometry data as well as to define processes that are executed by OpenVSP.
Custom components	If a specific user interface is needed or a special geometry that OpenVSP standard geometry parts do not cover, custom components can be applied. The parts are of the type *.vspscript and need to be saved in the folder */customcomponents* so that the main program has access [18].
XML parsing	A geometry part is saved in an XML format (*.vsp3) file [13].

In Tab. 4 eight different communication methods have been selected and evaluated concerning their level of automation and required workload.

Table 4. OpenVSP communication methods

	OpenVSP user interface	Command line batch mode	Design file (*.des)	Scripting (*.vspscript)	Custom components (*.vspscript)	XML parsing (*.vsp3)	Level of automation
M 1	✓		✓				+
M 2	✓	✓	✓				++
M 3	✓			✓	✓		+
M 4	✓					✓	+
M 5		✓	✓	✓			+++
M 6		✓		✓	✓		+++
M 7		✓		✓			+++
M 8	✓	✓	✓	✓	✓	✓	++

The workload is evaluated qualitatively by the three categories depicted below:

Required workload
High
Medium
Low

Each method can use one or multiple of the previous mentioned techniques. Due to this every method is analyzed and evaluated with equal significance of every individual technique.

It is valid for all methods that the OpenVSP geometry model files are of the type *.vsp3 and are built in an XML format. This enables techniques like XML parsing in which these simple text files can directly be manipulated and specific data can be changed. In order to visualize the files in the structured XML format Microsoft Visual Studio or any other integrated development environment (IDE) is required. However XML parsing is related to high workloads as many geometry parameters need to be changed. For that reason it is not meaningful and not recommended to use XML parsing for the development of an efficient communication standard.

Scripting code, which is written in AngelScript [18], parses the XML file by itself and includes more functionality than just parsing the *.vsp3 file. The batch mode is basically the same as running OpenVSP but without a user interface. Assuming that the geometry model is already present in ADEBO, a user interface of OpenVSP is not necessary in order to transfer data. For these reasons it is recommended to use method M7 which is a combination of Scripting and the OpenVSP batch mode. It suits the need for automation and also requires minimal workload. In order to maximize data consistency between ADDAM and OpenVSP custom components (see method M6) can be used. However this would lead to a higher workload.

The methods M1 - M4 require manual inputs into the graphical user interface which leads to less code to be written and fast results. Yet these methods are ruled out due to the fact that they are all semi-automatic in each communication cycle. Method M5 is a suitable alternative to method M7 with minor coding differences and can be used depending on personal preference.

Combining all techniques (reflected in method M8) shows that the techniques do not block each other and can be used with all their capabilities.

Finally it can be outlined that any program can be used to connect to OpenVSP as long as it has the capability to read and write *.txt files and the Windows® command window is accessible from within the program. If an integrated command window is missing an agent like a console window *.exe in Microsoft Visual Studio can be created in order to close this communication gap.

Figure 5 explicitly shows how method M7 is applied within ADEBO. Initially an ADDAM aircraft configuration with defined geometry is needed. Subsequently an export function extracts required geometry data and the *.vspscript file is created. Finally the batch mode is started which builds the *.vsp3 file and, if demanded, the user interface is started.

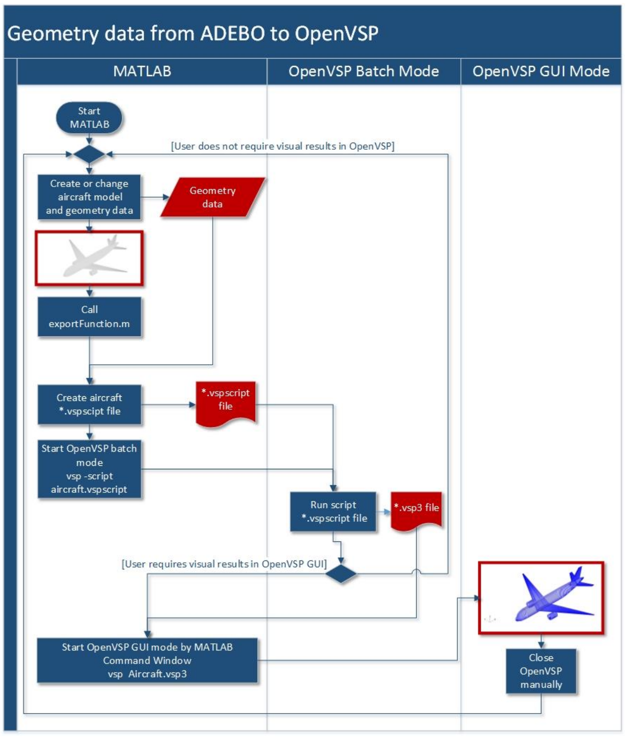
Geometry data from ADEBO to OpenVSP
MATLAB
OpenVSP Batch Mode
OpenVSP GUI Mode
Start MATLAB
[User does not require visual results in OpenVSP]
Create or change aircraft model and geometry data
Geometry data
Call exportFunction.m
Create aircraft *.vspscipt file
*.vspscript file
Start OpenVSP batch mode vsp -script aircraft.vspscript
Run script *.vspscript file
*.vsp3 file
[User requires visual results in OpenVSP GUI]
Start OpenVSP GUI mode by MATLAB Command Window vsp Aircraft.vsp3
Close OpenVSP manually

Figure 5. Export from ADEBO to OpenVSP

If an aircraft configuration is changed manually in OpenVSP or calculations have been performed the results can be transferred back into ADEBO with the import functionality, visualized in Fig. 6. For this purpose the respective ADDAM aircraft configuration has to be present in ADEBO. Initially the import function creates an *.vspscript file and starts the batch mode. Within the batch mode the *.vsp3 file, containing manipulated data by OpenVSP, is read and a *.txt file is created. In conclusion the new data is retrieved and replaces the old data in ADDAM.

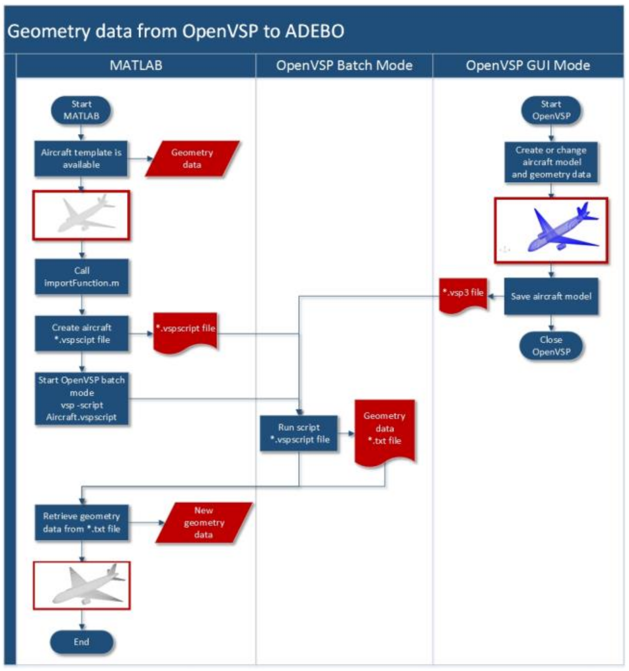
Geometry data from OpenVSP to ADEBO
MATLAB
OpenVSP Batch Mode
OpenVSP GUI Mode
Start MATLAB
Aircraft template is available
Geometry data
Call importFunction.m
Create aircraft *.vspscipt file
*.vspscript file
Start OpenVSP batch mode vsp -script Aircraft.vspscript
Run script *.vspscript file
Geometry data *.txt file
Retrieve geometry data from *.txt file
New geometry data
End
Start OpenVSP
Create or change aircraft model and geometry data
Save aircraft model
*.vsp3 file
Close OpenVSP

Figure 6. Import from OpenVSP to ADEBO

3.1 Requirements and boundary conditions

The software requirements for using ADEBO with OpenVSP are not highly demanding. It is recommended to use the MATLAB version R2014b or higher, otherwise minor errors might occur due to incompatibilities especially in visualization. OpenVSP is freely available at http://www.openvsp.org/ which provides the latest versions. The presented concept was developed using version 3.5.2. Table 4 summarizes the prevailing boundary conditions.

Table 4. Applied soft- and hardware

CPU	2.20 GHz
Memory (RAM)	8 GB
Operating System	Windows 7 64Bit
OpenVSP Version	v3.5.2
MATLAB Version	R2014b

Due to the open source availability of OpenVSP several files and concepts are already existing. The existing API provides all required coding needs [14] and no new source code had to be developed. There are also example scripts available like the *TestAll.vspscript* which provides a good starting point working with the AngelScript language and the OpenVSP coding. Furthermore OpenVSP contains multiple predefined geometry parts which are used for the export function. The advantage is that these parts are already very similar to the ADDAM geometry definitions. Additionally OpenVSP offers the option to build up new parts with their own geometry definition by creating custom components [15].

3.2 Implementation

The implementation comprised both developing MATLAB functions which create AngelScript files and building up process functions for a successful integration into ADEBO. The approximate workload amounts to 90 hours which corresponds to the skills of an advanced programmer in MATLAB and intermediate experience with using XML or AngelScript.
With minor challenges in finding the accurate OpenVSP variables during Scripting the API was successfully implemented. A helpful advice is to open an existing *.vsp3 file in e.g. Visual Studio and to search for the demanded variables. All problems that emerged during the implementation have been solved, although the solution for one problem needed additional effort: The *FUSELAGE* part included in OpenVSP could not be used. Instead the OpenVSP type *STACK* was applied which also resulted in a proper solution. Nevertheless the reason for this issue is addressed briefly:

The export code builds up the fuselage geometry cross section by cross section. Every parameter that defines a respective cross section is created and specified before moving on to the next set of parameters of the next cross section. Because the *Update()* command is called right after every parameter specification errors arise. The reason for this is that OpenVSP initially creates a body of type *FUSELAGE* with predefined x-positions of each cross section. During the adjustment of the x-positions according the ADDAM geometry, consecutive cross sections might be placed behind each other which would lead in a different cross section order. In the end this would result into moving beyond the x-position percentage boundaries.

3.2.1 Concept in MATLAB

The API concept in MATLAB is relatively simple. First of all there is a function which executes the command prompt within MATLAB using the *dos* function. Moreover there are import and export functions for each geometric definition of a part translating the data between MATLAB and OpenVSP by creating *.vspscript files using AngelScript. In the end the *.vspscript is executed in the command prompt which runs OpenVSP. The same procedure is done for all commands which shall be executed in OpenVSP. Altogether 21 *.m files are required in order to communicate and translate information between OpenVSP and MATLAB/ADEBO. Currently no parallelization in MATLAB was performed (see 3.3)

3.3.2 Realization in ADEBO

The API in MATLAB, described in section 3.3.2, needs to be integrated into the design environment ADEBO. This means an input and output wrapper, compare Fig. 7 is required.

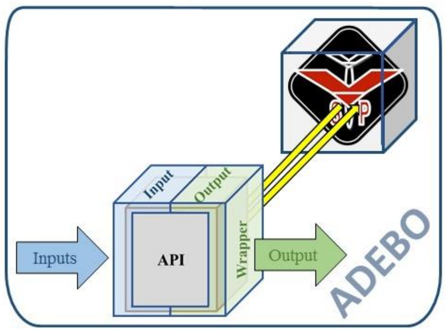

Figure 7. ADEBO wrapper concept

The input wrapper extracts the geometry data from ADDAM and checks them for completeness, correct size and unit. In case of a different unit than meters the value will be translated. Only if all data is valid the API is started, otherwise a warning message will stop the current process and indicate the reason. The output wrapper sets the resulting data which is transferred back or replaces changed geometry data. Besides tool specific information which characterize the last OpenVSP utilization is saved in ADDAM.

3.3 Workflow and capabilities

A complete automation in using OpenVSP as a tool within the design environment has been achieved. This incorporates the export and import of the geometry of the main aircraft components:

• Fuselage	*STACK*
• Lifting surface	*WING*
• Engine	*STACK*
• Motor	*STACK*
• Propeller	*STACK*

Moreover the fuselage can be defined using circular or elliptical cross sections and the lifting surface can be built up with as many partitions as necessary. Also the airfoil shape is considered. Regarding the freely selectable number of partitions and cross sections the required computational time has to be considered. On the one hand a high number may increase accuracy but on the other hand computational time increases exponentially (see Fig. 8).

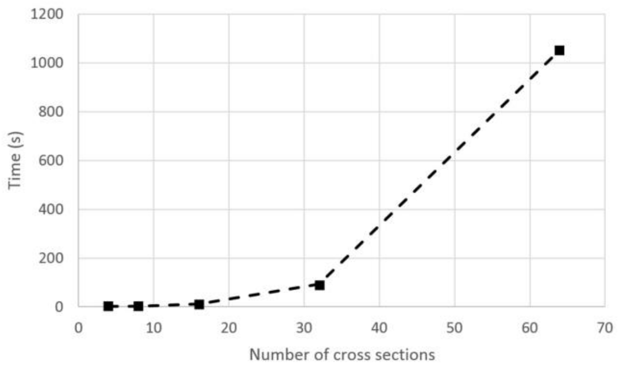

Figure 8. Export fuselage trade-off

The time ratio needed (see Fig. 5 Batch Mode) rises from 89.8% for the OpenVSP API and 10.2% MATLAB (4 cross sections) to 99.99% OpenVSP API and 0.01% MATLAB (64 cross sections). The bottleneck is probably caused by the OpenVSP computational performance boundaries.

Furthermore the wetted area of a component or an entire configuration can be calculated and *.stl files can be created automatically.

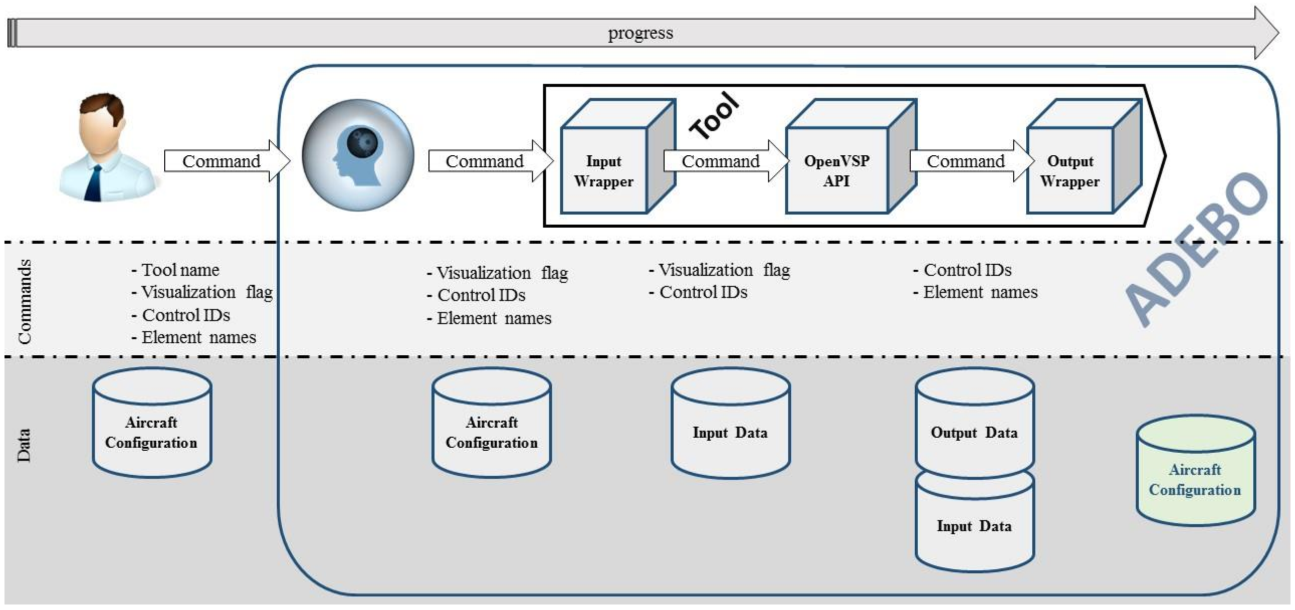

Figure 9. Workflow using OpenVSP in ADEBO

The typical workflow using OpenVSP in ADEBO is depicted in Fig. 9. The user commands the AE by defining the demanded tool name, a visualization flag (starting OpenVSP GUI or not) and by assigning the aircraft configuration object which shall be used. Optionally tool control parameters can be defined as well as specific component names in case the entire aircraft shall not be exported or imported. After these initial unified user commands all further process steps do not need any additional interventions. In the end all output data is automatically stored in the aircraft configuration.

4 Application of OpenVSP in a design process

OpenVSP can easily be used within a design schedule in ADEBO (cf. Fig. 2) which enables iterative or even optimization calculations. Also trade-off studies with e.g. different wing layouts can be performed by importing manipulated configurations from OpenVSP and using aerodynamic and performance calculation tools of ADEBO. A major advantage of OpenVSP is that it can be applied for all kinds of aircraft. The successfully proven test cases comprise a civil transport aircraft (A320), a military trainer (Alpha Jet) and also an RPAS with wingspan $b = 5\text{m}$ (IMPULLS). Figure 10 visualizes these aircraft successfully exported to OpenVSP.

A further advantage of applying OpenVSP in conceptual design phases is being able to calculate the wetted area of the aircraft with sufficient accuracy. Consequently the parasite drag prediction method introduced by Ramer [19] using a skin friction coefficient c_{fe} and the ratio of wetted aircraft area to wing reference area S_{wet}/S_{ref} can be enhanced. The calculation

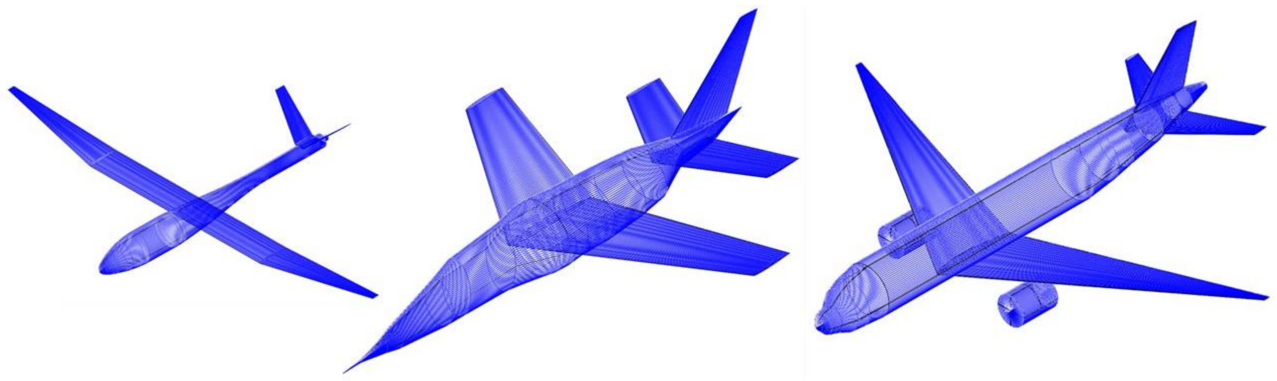

Figure 10. Imported aircraft in OpenVSP (IMPULLS, Alpha Jet, A320)

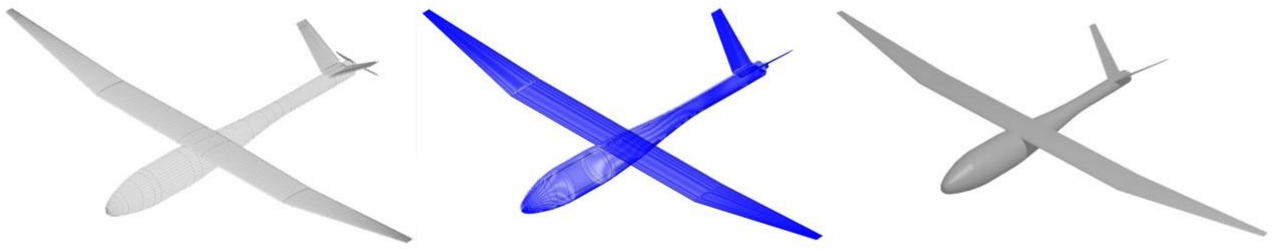

Figure 11. Visualization models of IMPULLS (ADDAM, OpenVSP, *.stl in Blender®)

of the area ratio, which may be a source of error especially for configurations which deviate from the pictured aircraft in [19], does not need to be estimated anymore. The difference of an estimated S_{wet}/S_{ref} for an aircraft similar to a A320 is:

- $S_{wet}/S_{ref} = 6$ (estimated)
- $S_{wet}/S_{ref} = 5.6$ (calculated)

Although it is only an accuracy increase of 6.66%, the application of a physics-based method in early phases of the design process can contribute to achieve more precise results and a more efficient aircraft.

Besides created *.stl files can be exploited for initial CAD developments and also as visualization input for simulations. Figure 11 shows the consistency of all three visualization models started from ADDAM to OpenVSP to a *.stl file in Blender®.

5 Summary and outlook

The presented work shows the successful development of an API for communication and data transfer between MATLAB and OpenVSP. Hence OpenVSP is integrated as new tool element in the aircraft design environment ADEBO. The geometry data of lifting surfaces, fuselages, engines, motors and propellers can not only be exported from ADEBO to OpenVSP, but also imported. This allows easy geometry manipulation by using the OpenVSP GUI and transferring back to the data model ADDAM. Also calculation capabilities like determining the wetted area can be used. Furthermore *.stl files of the aircraft configuration or single components can be created automatically. All in all OpenVSP enhances the design capabilities of ADEBO and is suitable for iterative or optimization calculations.

A major advantage of OpenVSP is its open source availability and its uncomplicated API access, which simplifies working on it. The API for communication can be specifically tailored for the needs of using it within ADEBO and changes in the still ongoing development can be made easily.

The next development steps consist of adding custom components in order to increase user-friendliness and performing some smaller code optimizations which can be done for example in the *Update()* commands. Finally one improvement suggestion for further OpenVSP versions would be to provide functionalities to access an already running OpenVSP batch mode or to allow scripts to run consecutively by keeping a single batch mode open. This would reduce computation time because the *.vsp3 geometry file would not always have to be reloaded after a single *.vspscript file has finished its computation.

References

[1] Sobieski, J., *Recent experiences in multidisciplinary analysis and optimization. Proceedings of a symposium held at NASA Langley Research Center, Hampton, Virginia, April 24-26, 1984*, National Aeronautics and Space Administration, Scientific and Technical Information Branch; [National Technical Information Services, distributor], [Washington, D.C.], [Springfield, Va.], 1984.

[2] Ahmad, S., and Thomas, K., "Flight Optimization System (FLOPS) Hybrid Electric Aircraft Design Capability," *The Spectra*, 2013, pp. 44–49.

[3] Nagel, B., Böhne, D., Gollnick, V., Schmollgruber, P., Rizzi, A., La Rocca, G., and Alonso, J., "Communication in Aircraft Design: Can we Establish a Common Language?," *ICAS Congress*

of the International Council of the Aeronautical Sciences, 2012.

[4] Seider, D., Litz, M., Schreiber, A., Fischer, P. M., and Gerndt, A., “Open Source Software Framework for Applications in Aeronautics and Space,” *2012 IEEE Aerospace Conference*, IEEE, Piscataway, NJ, 2012, pp. 1–11.

[5] Ziemer, S., Glas, M., and Stenz, G., “A Conceptual Design Tool for Multi-Disciplinary Aircraft Design,” *2011 IEEE Aerospace Conference*, 2011, pp. 1–13.

[6] Lukaczyk, T. W., Wendorff, A. D., Colonno, M., Economon, T. D., Alonso, J. J., Orra, T. H., and Ilario, C., “SUAVE: An Open-Source Environment for Multi-Fidelity Conceptual Vehicle Design,” *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2015.

[7] Moore, M., “Vehicle Sketch Pad | User Manual,” NASA Langley Research Center.

[8] “MATLAB - MathWorks - MathWorks Deutschland,” http://de.mathworks.com/products/matlab/, [retrieved 10 May 2016].

[9] Herbst, S., and Hornung, M., “ADDAM: An Object Oriented Data Model for an Aircraft Design Environment in MATLAB,” *AIAA Modeling and Simulation Technologies Conference*, 2015.

[10] Herbst, S., “ADEBO: Aircraft Design Box,” Institute of Aircraft Design, TUM, 25 Apr. 2016.

[11] Herbst, S., “ADDAM: Aircraft Design Data Model,” Institute of Aircraft Design, TUM, 25 Apr. 2016.

[12] “OpenVSP 3.0 API,” www.openvsp.org/wiki/lib/exe/fetch.php?media=jr_openvsp30_api.pdf, [retrieved 17 May 2016].

[13] “Script/API Functions And Classes [OpenVSP],” http://www.openvsp.org/wiki/doku.php?id=apiusecase, [retrieved 17 May 2016].

[14] “API [OpenVSP],” http://www.openvsp.org/wiki/doku.php?id=api, [retrieved 17 May 2016].

[15] Gloudemans, J. R., and McDonald, R., “User Defined Components In The OpenVSP Parametric Geometry Tool,” www.openvsp.org/wiki/lib/exe/fetch.php?media=workshop15:custom_components_aviation_2015.pdf, [retrieved 17 May 2016].

[16] McDonald, R., “VSP Design Files & XDDM for Cart3D Optimization,” www.openvsp.org/wiki/lib/exe/fetch.php?media=xddm_2013.pdf, [retrieved 17 May 2016].

[17] McDonald, R., “VSP Design Files & XDDM for Cart3D Optimization,” www.openvsp.org/wiki/lib/exe/fetch.php?media=workshop15:vsp_xddm.pdf, [retrieved 17 May 2016].

[18] “Custom Component [OpenVSP],” http://www.openvsp.org/wiki/doku.php?id=custom, [retrieved 17 May 2016].

[19] Raymer, D. P., *Aircraft design. A conceptual approach*, 5th edn., American Institute of Aeronautics and Astronautics, Reston, VA, 2012.

Copyright Statement