GEOMETRY EXPORT FROM OPENVSP TO CPACS

Kagan Atci

2025 OpenVSP Workshop
July 10, 2025

German Aerospace Center (DLR e.V.)
Institute of Flight Systems

DLR

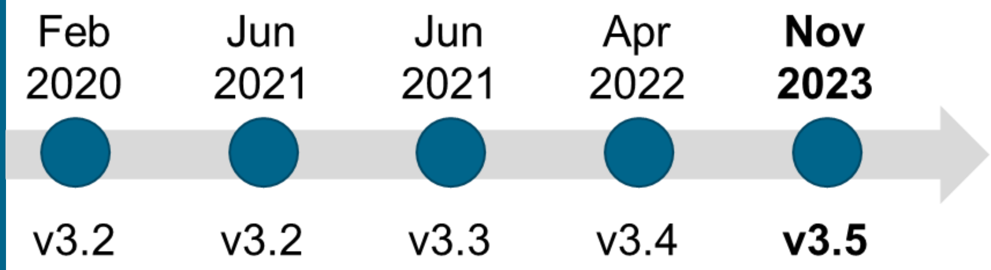
Outline

- CPACS
  - Overview
  - Data structure
- VSP2CPACS
  - Overview
  - Wing export
  - Fuselage export
  - Rotor export
- Studies and examples
- Future work

CPACS
Overview

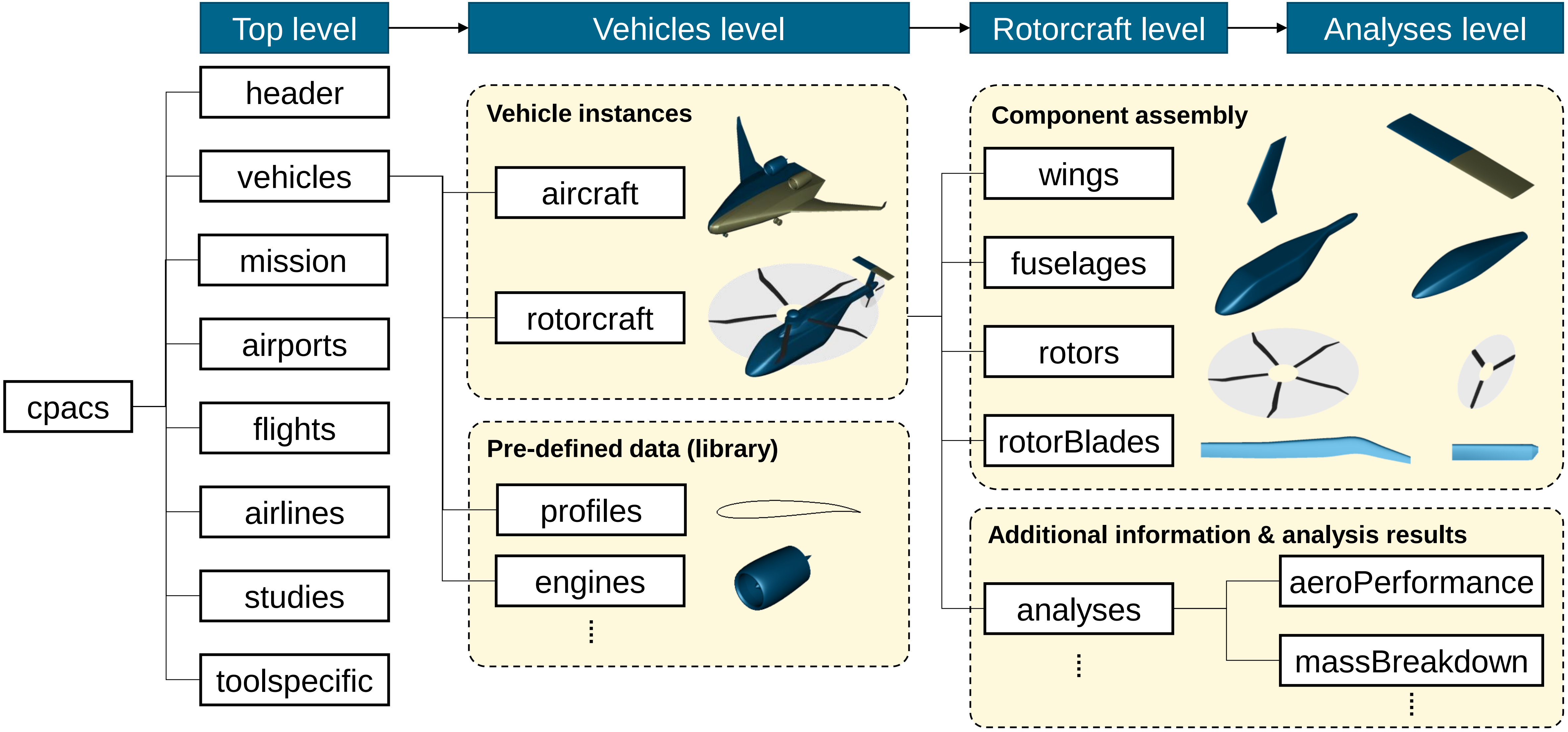
CPACS – Common Parametric Aircraft Configuration Schema

- Central data exchange format for multidisciplinary aircraft design and analysis
- Development led by DLR Institute of System Architectures in Aeronautics
- Aircraft data is stored in form of a hierarchical data structure in XML
- Widely used in interdisciplinary collaborations across multiple projects within DLR



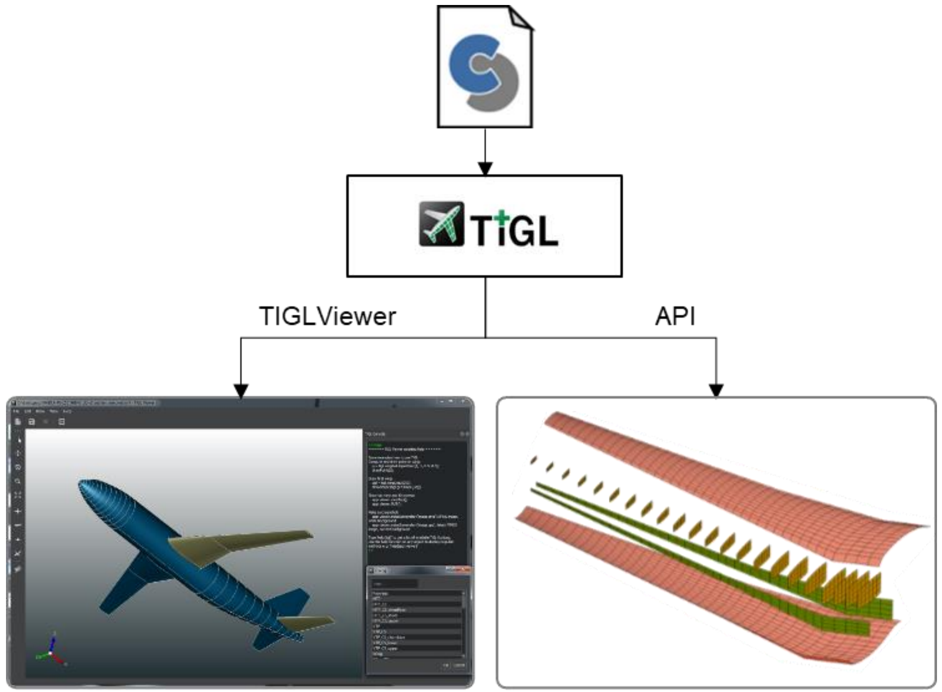
Marko Alder, Erwin Moerland, Jonas Jepsen, Björn Nagel
Recent Advances in Establishing a Common Language for Aircraft Design with CPACS
Aerospace Europe Conference 2020
Feb 25-28, 2020, Bordeaux, France

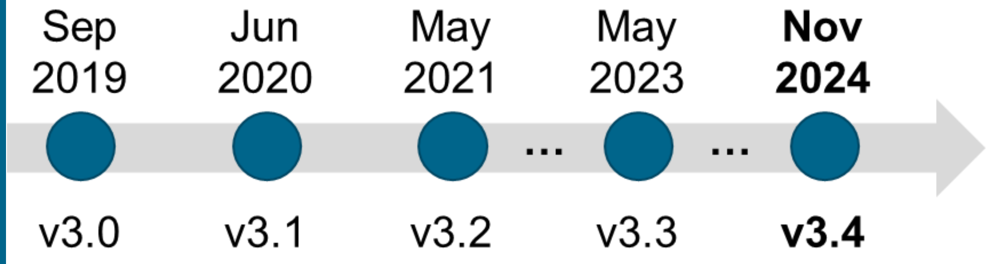
CPACS
Hierarchical data structure



TiGL Geometry Library TiGL

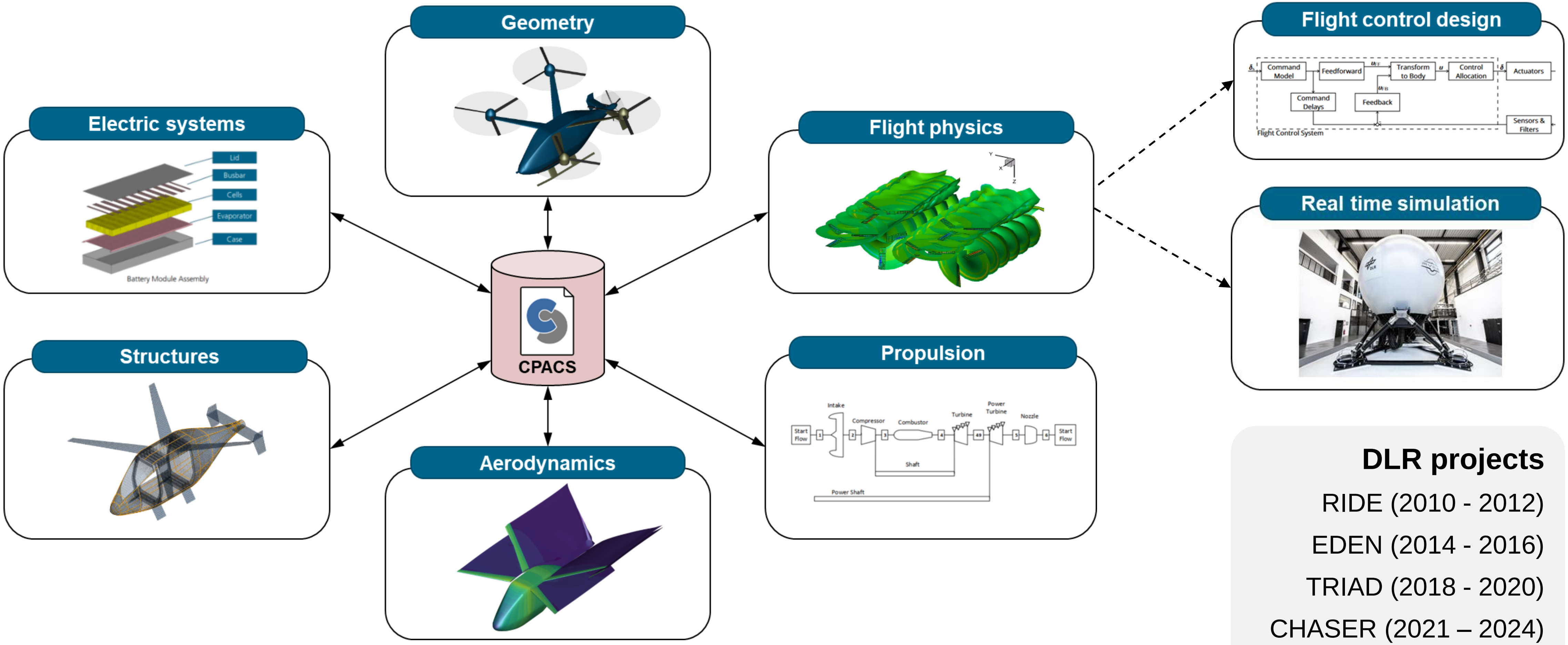
- Developed for creating 3D geometric representation of the parametric geometry data from CPACS files
- Uses OpenCASCADE CAD kernel (NURBS surfaces)
- API: high level functions for various geometry calculations (e.g. point query, surface area, shape volume, intersection)
- TiGLViewer: geometry display interface





Martin Siggel, Jan Kleinert, Tobias Stollenwerk, Reinhold Maierl
TiGL: An Open Source Computational Geometry Library for Parametric Aircraft Design
Mathematics in Computer Science (2019) 13:367-389

CPACS
Applications in rotorcraft design



DLR projects

RIDE (2010 - 2012)

EDEN (2014 - 2016)

TRIAD (2018 - 2020)

CHASER (2021 – 2024)

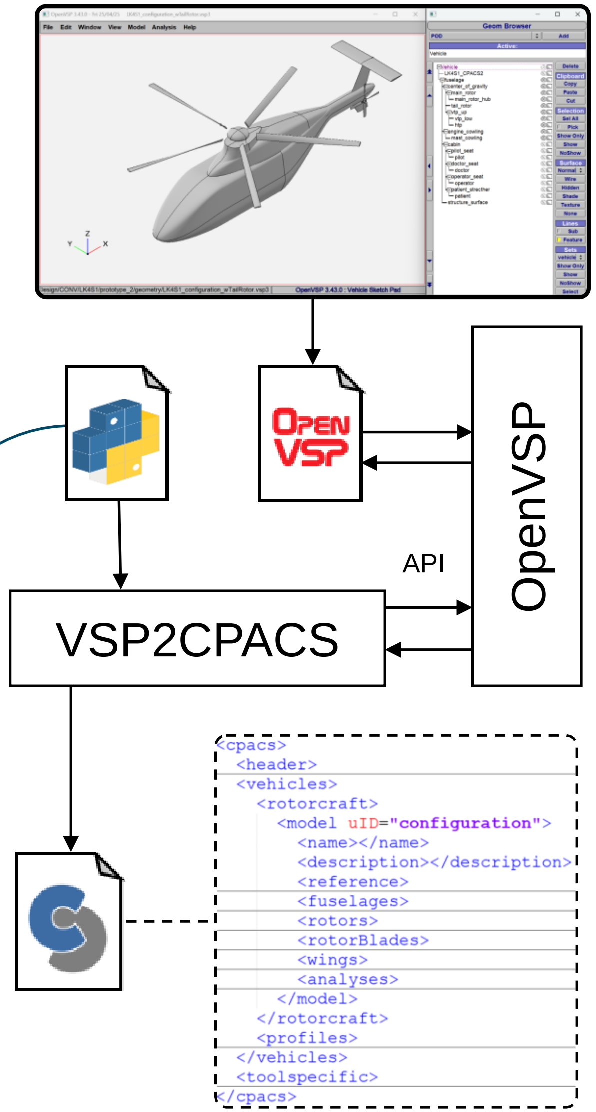
ARCADE (2025 - 2028)

VSP2CPACS

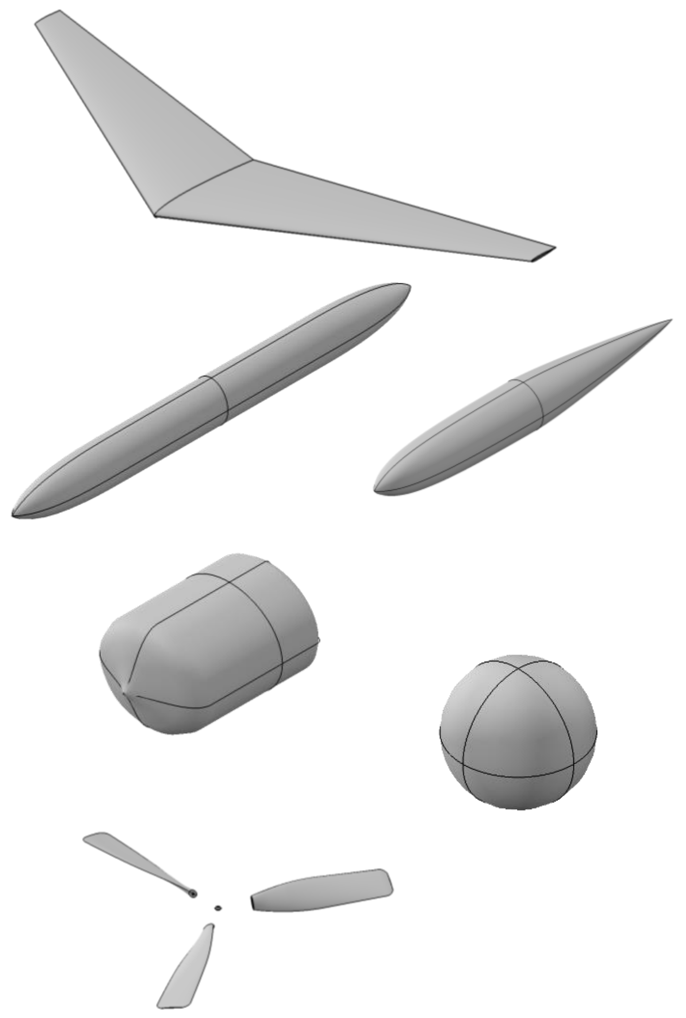
- **Motivation:** Initializing CPACS files for generic aircraft / rotorcraft concepts containing the basic geometry.
- **VSP2CPACS** is a python package developed to export geometry data from OpenVSP to CPACS
  - Uses OpenVSP API to access the geometry
  - Current versions: OpenVSP **v3.43.0** to CPACS **v3.4**
  - Process is initialized with a python script

```
import vsp2cpacs as v2c
# Set path
v2c.setVSP3DirectoryPath(...)
v2c.setVSP3FileName(...)
v2c.setCPACSDirectoryPath(...)
v2c.setCPACSFileName(...)
# Select geoms
v2c.addGeomElement(GEOM_NAME='Fuselage', CPACS_UID='fuselage')
v2c.addGeomElement(GEOM_NAME='MainRotor', CPACS_UID='mainRotor')
v2c.addGeomElement(GEOM_NAME='HTail', CPACS_UID='htp')
⋮
v2c.run()
```

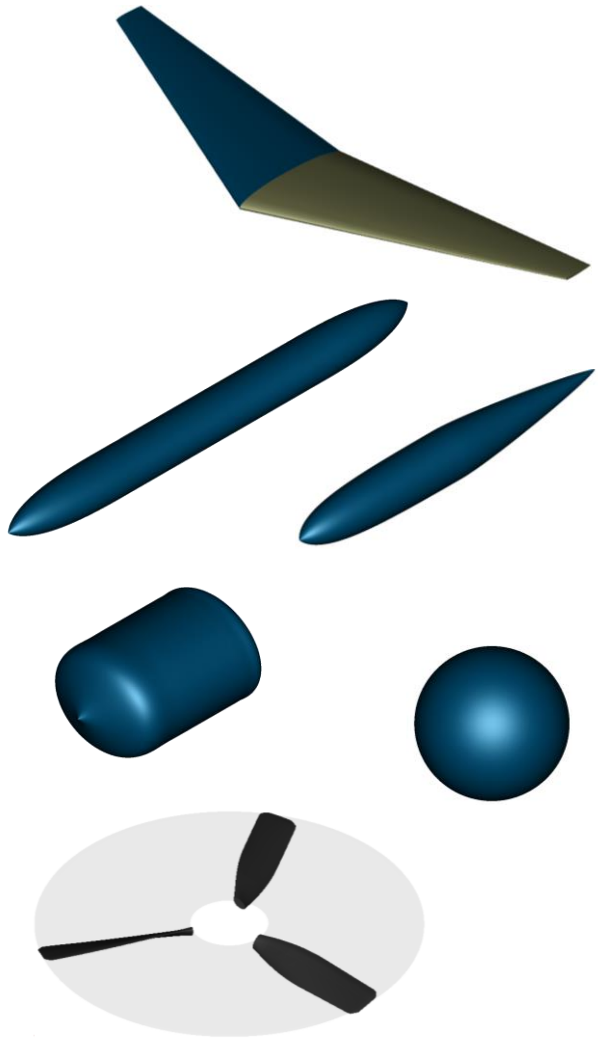
  - XML file is written featuring the selected geoms in CPACS format



VSP2CPACS
Compatible geoms

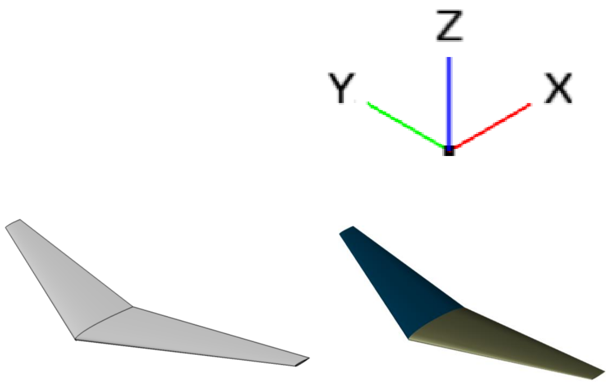


OpenVSP geom	CPACS element
→ Wing	<wing> ←
→ Fuselage → Stack → Pod → Conformal → Ellipsoid → BOR (limited)	<fuselage> ←
→ Prop	<rotor> ← <rotorBlade> ←

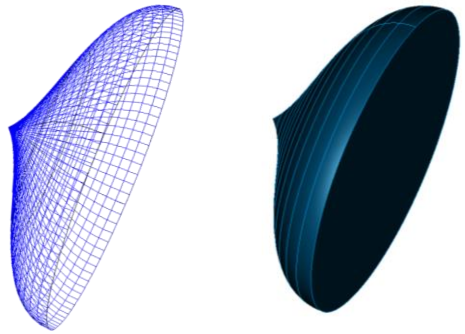


VSP2CPACS
Commonalities and conditions

- Commonalities between OpenVSP and CPACS
  - Coordinate system (Euler angles: x → y‘ → z“)
  - Similar wing, fuselage and rotor parameterization (to a limited extent)
  - Section and profile definitions (to a limited extent)
  - Symmetry plane definitions



- Conditions
  - CPACS element identification by uID attributes
    - OpenVSP Geoms must have unique names
  - Monotonic design policy
    - Sections must feature discrete spacing between each other
  - Shape ending with hollow sections or points
    - Hollow sections ends are filled in TiGL
    - No root / tip caps or flat end surfaces in geoms



VSP2CPACS
Geometry export

- Extraction sequence of lofted surfaces
  - Lofted geometry → `<wing>`, `<fuselage>`, `<rotorBlade>`
  - Read explicit parametric data (e.g. translation, rotation)
  - Read unrotated geometry

    1. Duplicate geom (separate from all dependencies)

    2. Set absolute location and rotation components to 0.

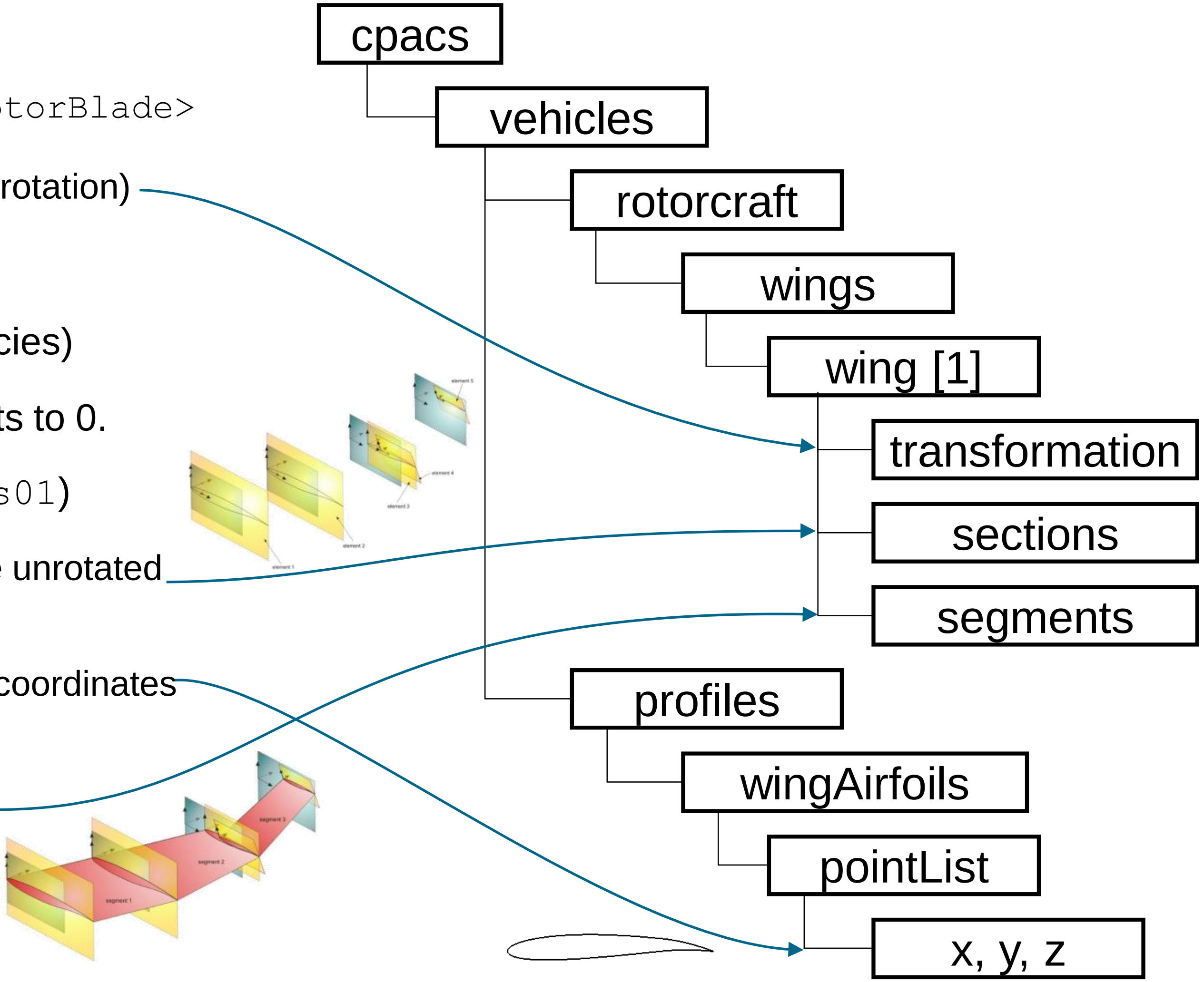
    3. Loop over all U panel (`openvsp.GetUWTess01`)

      3.1. Read section reference coordinates of the unrotated geom at each U panel

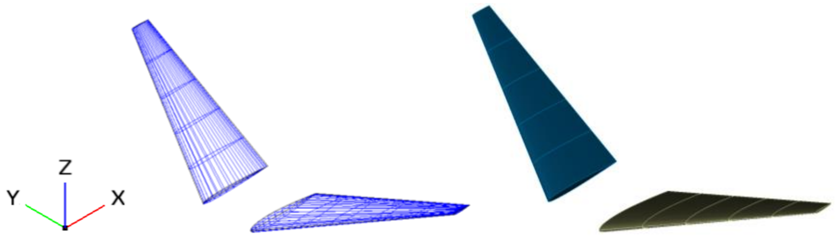
      3.2. Extract profile points at U w.r.t. reference coordinates (`openvsp.CompPnt01`)

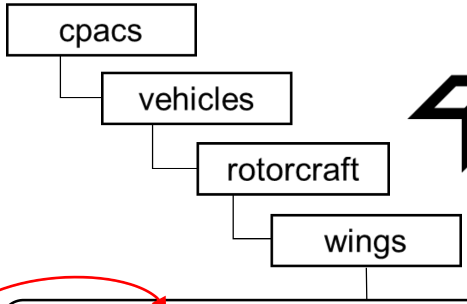
    4. Build segments → parametric connection of consecutive sections for surface lofting

    5. Delete copied geom

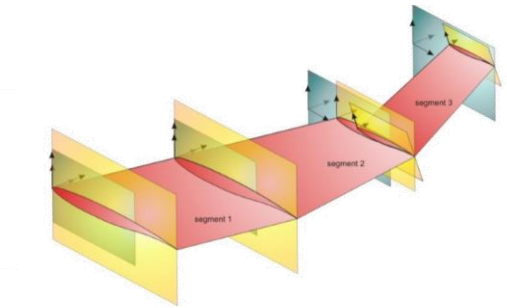


VSP2CPACS
Wing Geom → <wing>

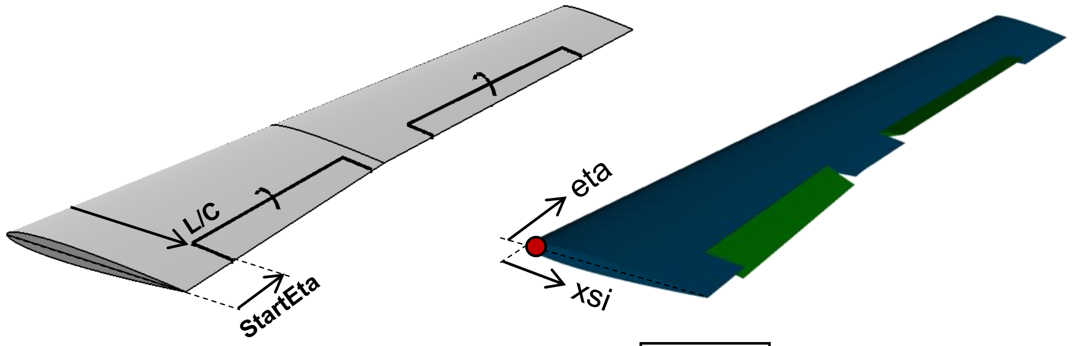




- Explicit parameters: Name, translation, rotation, and symmetry plane
- `<section>` transformation data of the reference plane for the profile
  - Location at the profile leading edge (`openvsp.CompPnt01`)
  - No rotation → profiles are extracted as 3D
  - Sum of U tessellations from all Sects
- <segment> → parametric connection of two consecutive section elements
  - One less than the number of `<section>`s

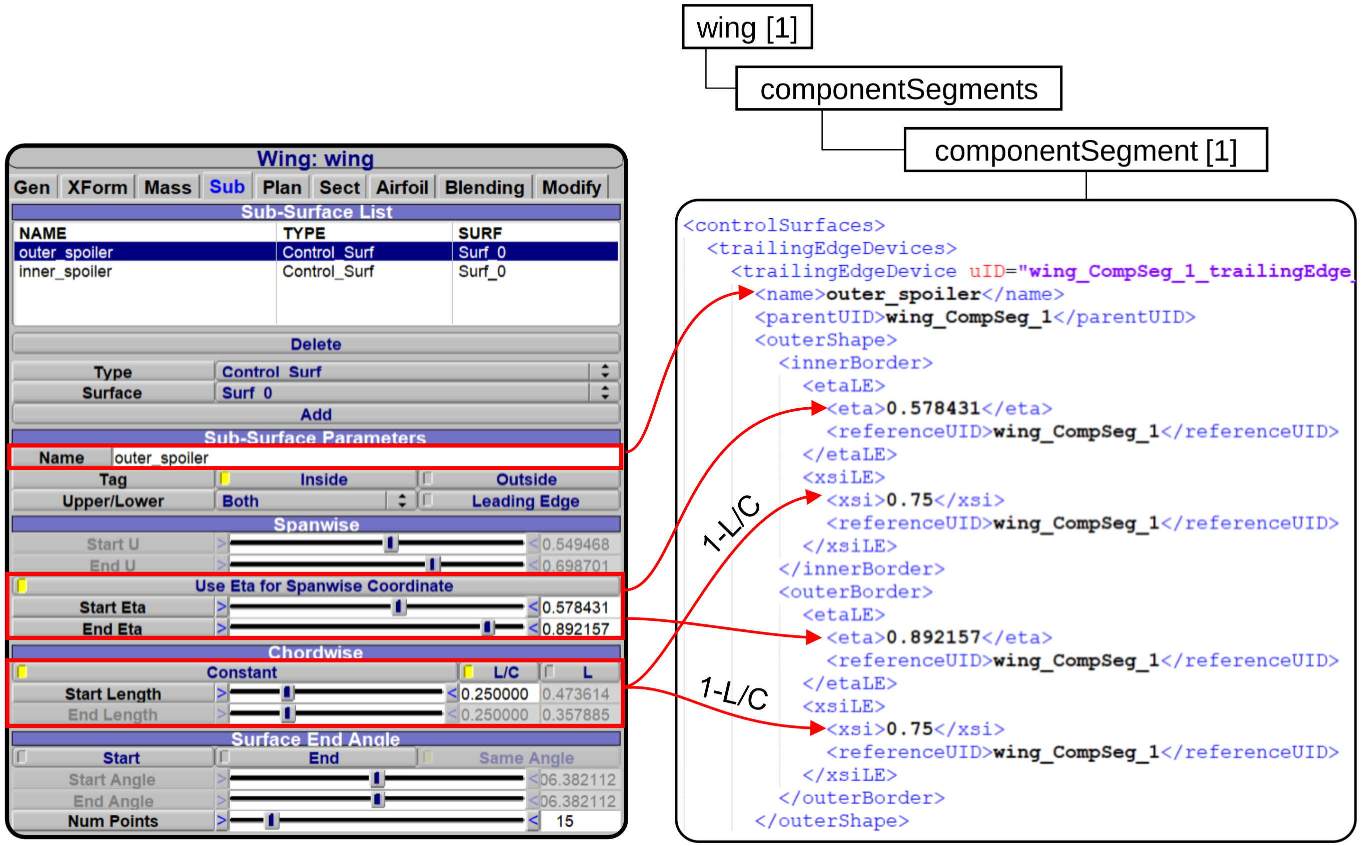


VSP2CPACS
Wing Geom → <wing>

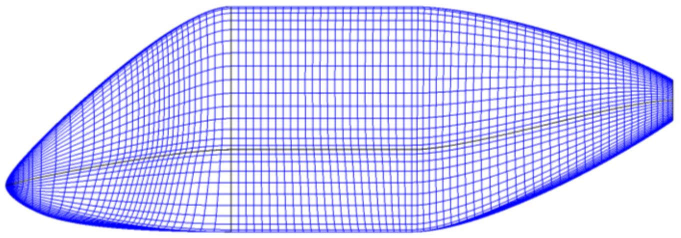


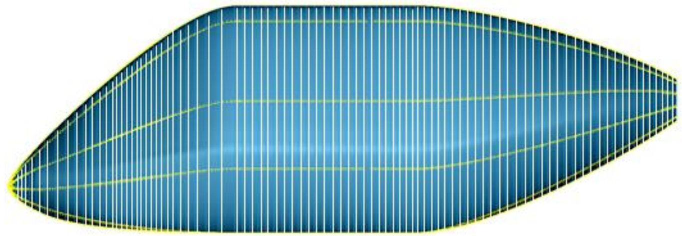
Control surfaces

- Definition under `<componentSegments>`
- Geometry export as explicit data
  - Name → Suffix in `uID` and `<name>`
  - Outer shape in form of rel. coordinates
    - **Start Eta** → `innerBorder/etaLE`
    - **End Eta** → `outerBorder/etaLE`
    - **L/C** → `<xsiLE>, <xsi>` = 1-L/C
  - Current limitations
    - Constant chordwise
    - Trailing edge only
    - No end angles
- Surface deflections have to be edited in the CPACS file following geometry export



VSP2CPACS
Fuselage Geom → <fuselage>





- Lofting procedure similar to <wing>
- Profile points are to be defined on a planar surface
- Planar slicing of the fuselage geom

  1. Get reference points at U (RST coordinates)
  Section center point $\mathbf{P_0}$ = `openvsp.CompPntRST(r=U, s=.5, t=0.5)`
  Section upper point $\mathbf{P_U}$ = `openvsp.CompPntRST(r=U, s=.5, t=0.0)`
  Section upper point $\mathbf{P_L}$ = `openvsp.CompPntRST(r=U, s=.5, t=1.0)`

  2. Point: $\Delta z = 0$ & $\Delta x = 0$ → no planar slicing

  3. Create normal vector $\mathbf{T}$ of the reference plane at $\mathbf{P_0}$

  4. Loop over all W coordinates

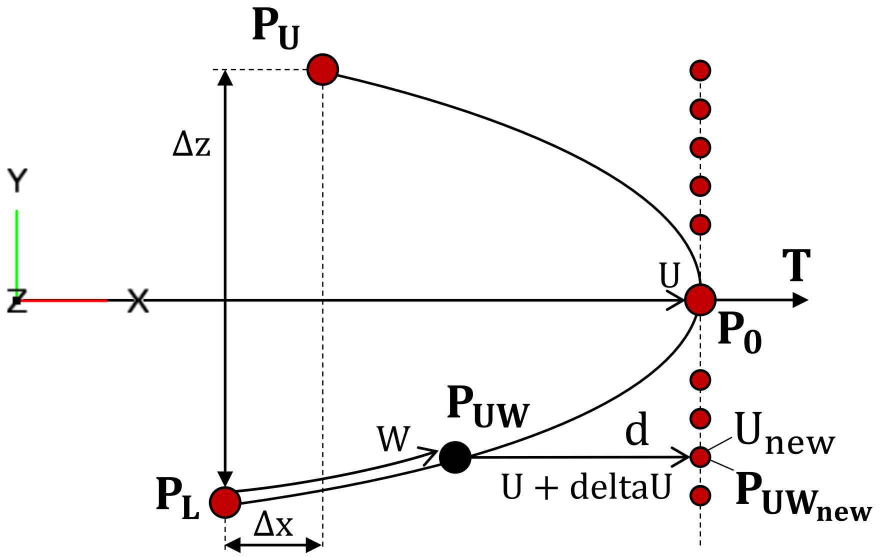
    4.1 Iterate U at constant W: $\mathrm{U_{new}} = \mathrm{U} + \mathrm{deltaU}$

    4.2 Get point $\mathbf{P_{UW_{new}}}$ at $\mathrm{U_{new}}$ (`openvsp.CompPnt01`)

    4.3 Calculate planar distance $\mathrm{d} = \mathbf{T} \cdot (\mathbf{P_0} - \mathbf{P_{UW_{new}}})$

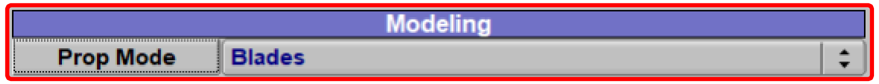
    4.4 If $\mathrm{d} <= \mathrm{tolerance}$ → $\mathbf{P_{UW}} = \mathbf{P_{UW_{new}}}$

    4.5 Overshoot: $\mathrm{d}/\mathrm{d_{Prev}} < 0$
    → Repeat the sequence with $\mathrm{deltaU_{new}} = -\mathrm{deltaU}/2$

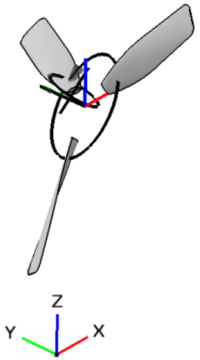


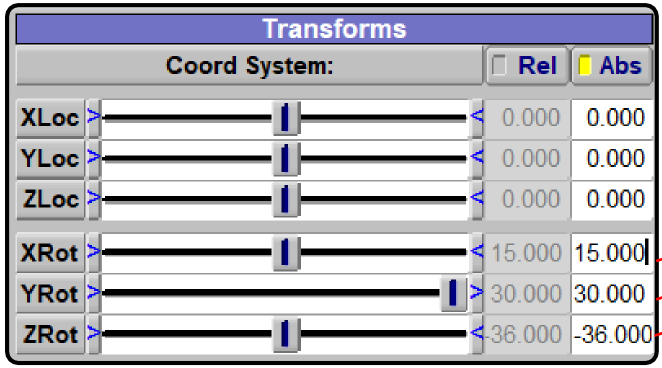
VSP2CPACS
Propeller Geom → <rotor>

- Parametric data is stored in `<rotor>`
  - Rotor translation, rotation and number of blades are taken as numbers from Propeller Geom
- Blade geometry is stored in `<rotorBlade>`
  - Lofting procedure identical to `<wing>`
  - First blade from the Propeller Geom

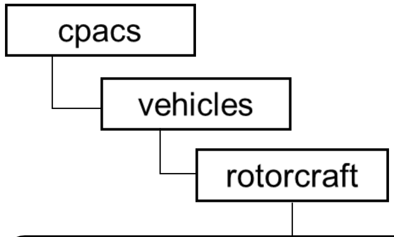


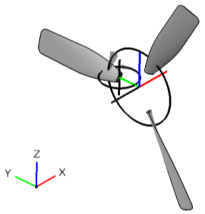
- Different rotor orientation in CPACS






```xml
<rotors>
  <rotor uID="rotor">
    <name>rotor</name>
    <transformation>
      <rotation>
        <x>-40.4646</x>
        <y>-44.4775</y>
        <z>0</z>
      </rotation>
    </transformation>
```





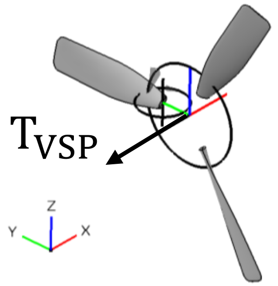
```xml
<rotors>
  <rotor uID="rotor">
    <name>rotor</name>
    <description></description>
    <nominalRotationsPerMinute>0</nominalRotationsPerMinute>
    <transformation>
      <rotation>
        <x>-0</x>
        <y>-90</y>
        <z>0</z>
      </rotation>
    </transformation>
    <rotorHub>
      <rotorBladeAttachments>
        <rotorBladeAttachment>
          <numberOfBlades>3</numberOfBlades>
          <rotorBladeUID>rotor_Blade</rotorBladeUID>
          <hinges/>
        </rotorBladeAttachment>
      </rotorBladeAttachments>
    </rotorHub>
  </rotor>
</rotors>
<rotorBlades>
  <rotorBlade uID="rotor_Blade">
</rotorBlades>
```

VSP2CPACS

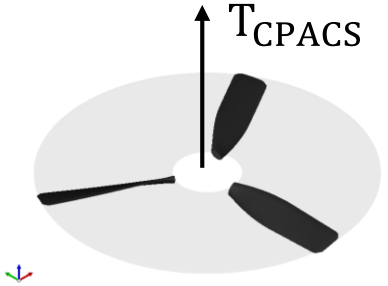
Propeller Geom → <rotor>

- Default thrust vectors

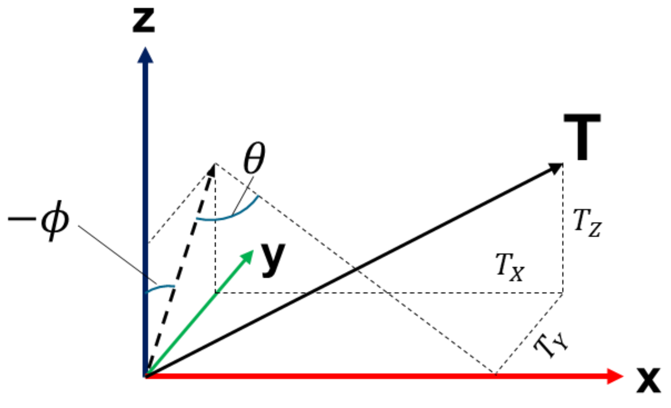
$\mathrm{T_{VSP}} = [-1 \quad 0 \quad 0]$
(Airplane propeller)



$\mathrm{T_{CPACS}} = [0 \quad 0 \quad 1]$
(Helicopter rotor)



- Thrust vector transformation between OpenVSP and CPACS
  1. Calculate the thrust vector $\mathbf{T} = \mathrm{R_Z}\,\mathrm{R_Y}\,\mathrm{R_X}\,\mathrm{T_{VSP}}$ (using `vec3d()` and `Matrix4d()` from `openvsp`)
  2. Calculate rotation angles ϕ and θ w.r.t. $\mathrm{T_{CPACS}}$



$$\mathbf{T} = [T_X \quad T_Y \quad T_Z]^T$$

$$\phi = -\operatorname{atan}\left({}^{T_Y}/_{T_Z}\right)$$

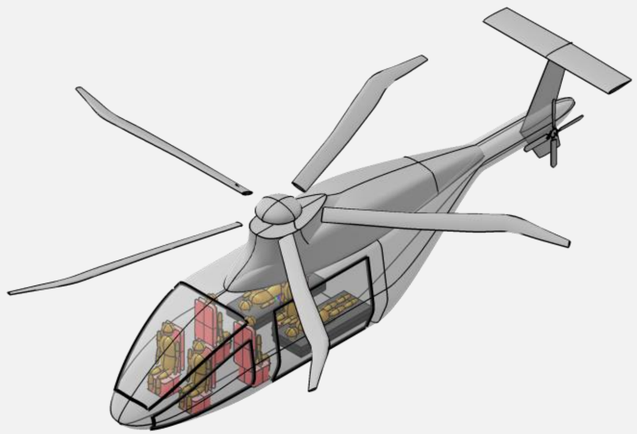
$$\theta = \operatorname{atan}\left(\frac{T_X}{\sqrt{T_Y^2 + T_Z^2}}\right)$$

```xml
<rotors>
  <rotor uID="rotor">
    <name>rotor</name>
    <transformation>
      <rotation>
        <x>-40.4646</x>
        <y>-44.4775</y>
        <z>0</z>
      </rotation>
    </transformation>
```

Studies and examples
DLR rotorcraft concepts

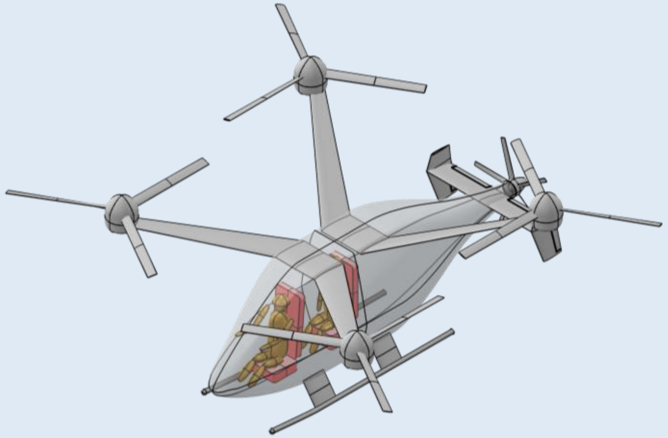
DLR

Primary Rescue Helicopter (PRH)



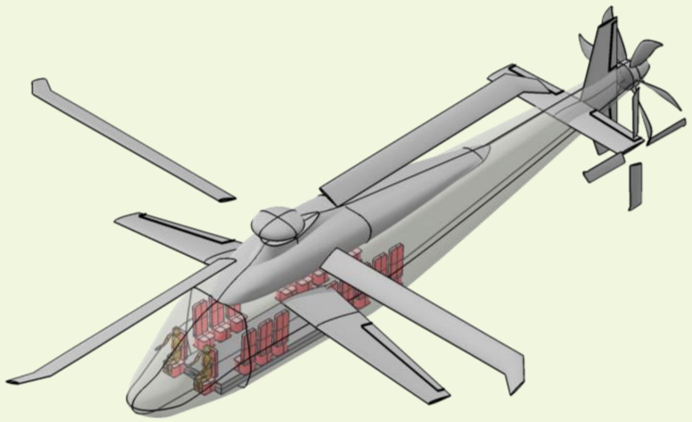
- Comprehensive design study for HEMS
- Conventional main rotor / tail rotor system
- Design speed: 300 km/h

Emergency Doctor Shuttle (EDS)



- Emergency physician transport
- Quadrotor system with pusher propeller
- Design speed: 150 km/h

High Speed Rotorcraft (HSR)



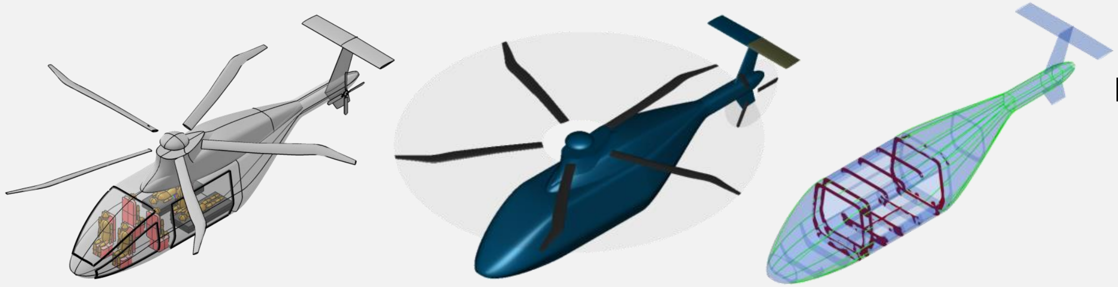
- Medium utility rotorcraft for high speeds
- Compound helicopter with pusher and wing
- Design speed: 450 km/h

Peter Weiand, Kagan Atci, et al.
Development of a New Fleet Concept for HEMS Operations
Vertical Flight Society's 81st Annual Forum & Technology Display
May 20-22, 2025, Virginia Beach, VA

Peter Weiand, Kagan Atci, Dominik Schwinn
High-Speed Compound Rotorcraft Performance and Design
Vertical Flight Society's 80th Annual Forum & Technology Display
May 7-9, 2024, Montreal, Canada

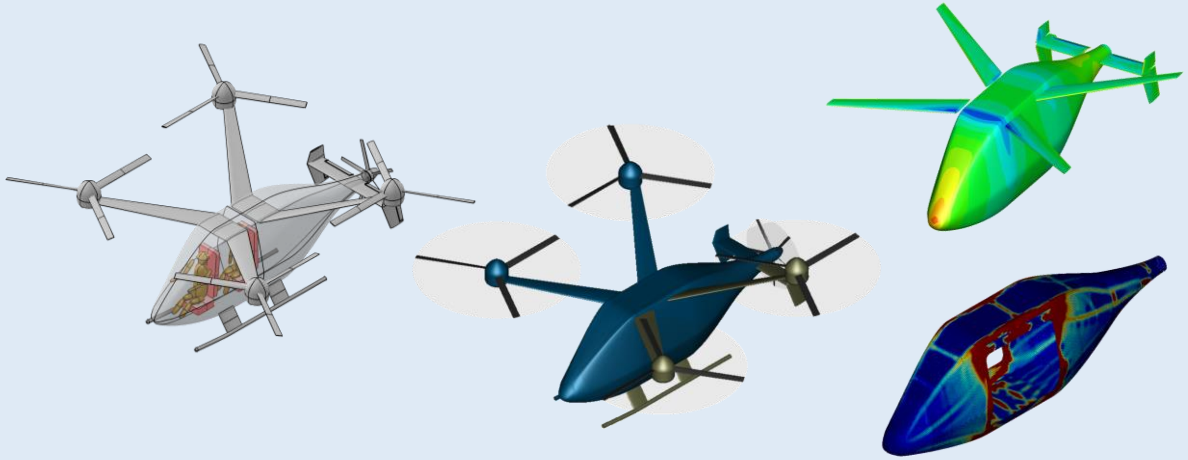
Studies and examples
DLR rotorcraft concepts

PRH



Dominik Schwinn, Michael Petsch, Dieter Kohlgrüber, Peter Weiand, Kagan Atci
Topology Optimization of Rotorcraft Airframes in Early Design Stages
Vertical Flight Society’s 81st Annual Forum & Technology Display
May 20-22, 2025, Virginia Beach, VA

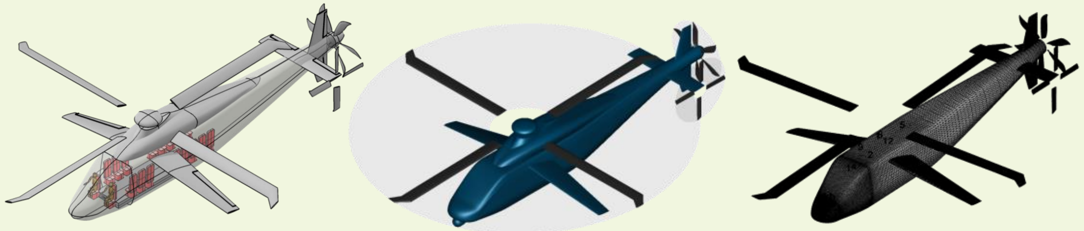
EDS



Kagan Atci, Peter Weiand, Hilal Inac
An Initial Flight Performance Assessment of New Urban Air Mobility Concept Vehicles for Medical Rendezvous Operations
Vertical Flight Society’s 81st Annual Forum & Technology Display
May 20-22, 2025, Virginia Beach, VA

Dominik Schwinn, Kagan Atci, Peter Weiand, Dieter Kohlgrüber
Structural Optimization of eVTOL Airframes
2nd International Conference on Advanced Air Mobility Systems (ICAAMS)
Dec 4-6, 2024, Singapore

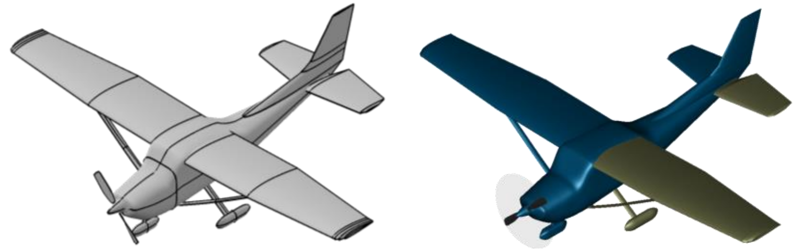
HSR



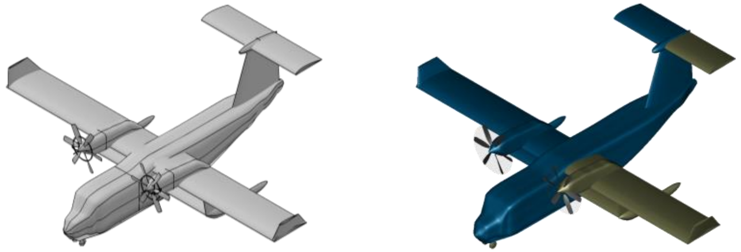
Johannes Wartmann, Peter Weiand, Kagan Atci
High-Speed Compound Rotorcraft Flight Control Design and Assessment
To be presented in 51st European Rotorcraft Forum
Sep 9-12, 2025, Venice, Italy

Applications and examples
VSP Airshow

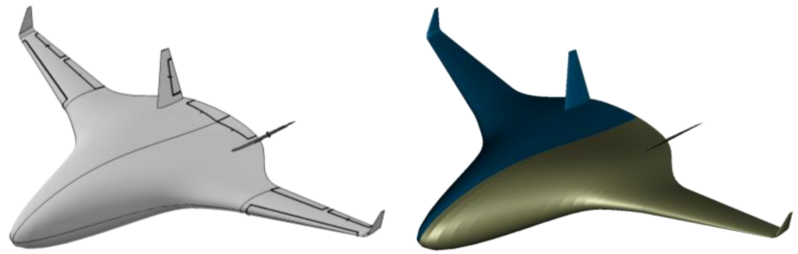
C172



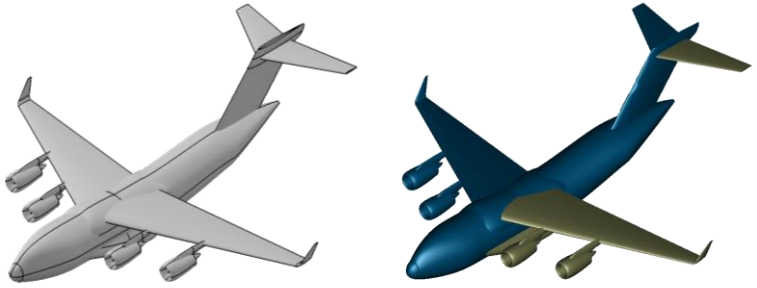
CL-415



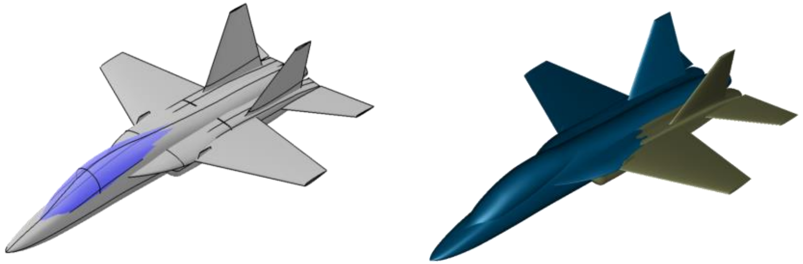
BWB



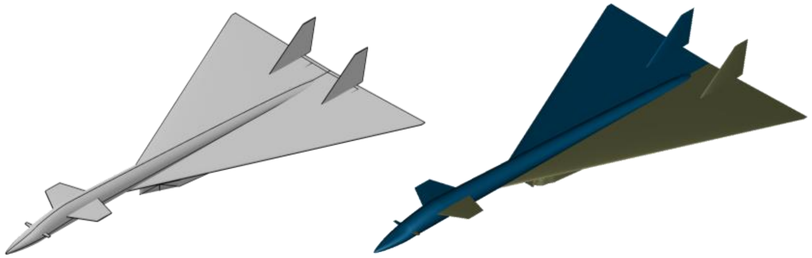
C-17



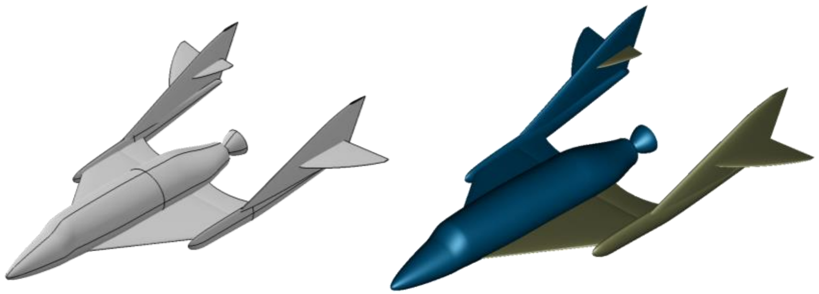
T-7A



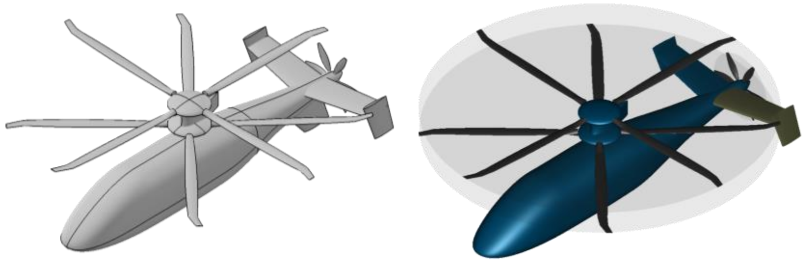
XB-70



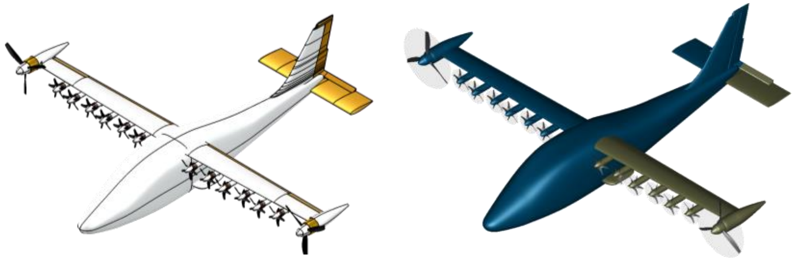
SpaceShipTwo



S-97 Raider



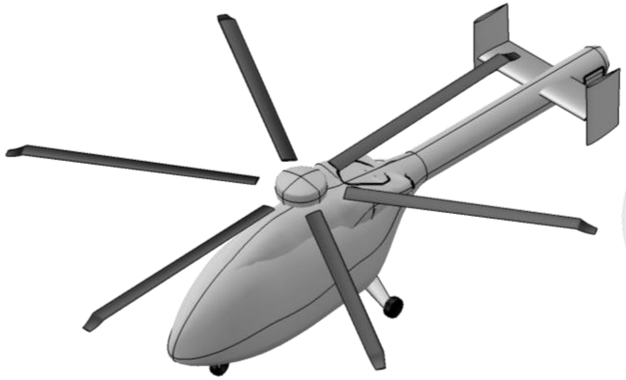
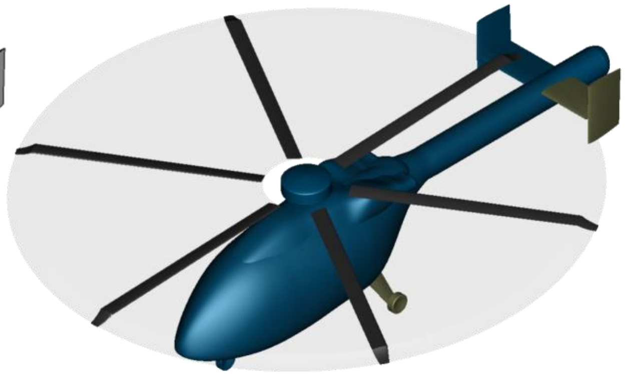
X-57 Maxwell



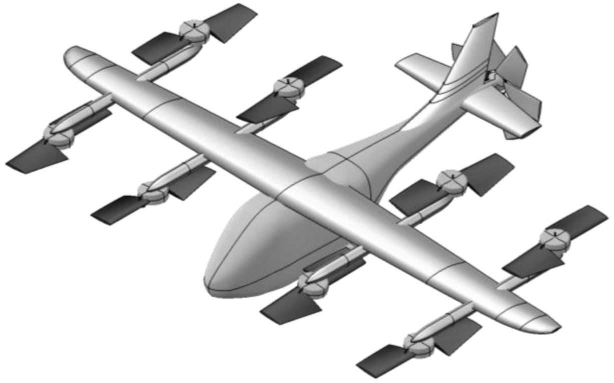
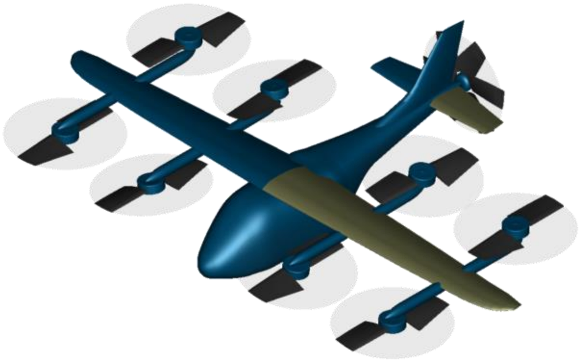
Models at https://airshow.openvsp.org/

Applications and examples
NASA Six-Passenger UAM Reference Vehicles

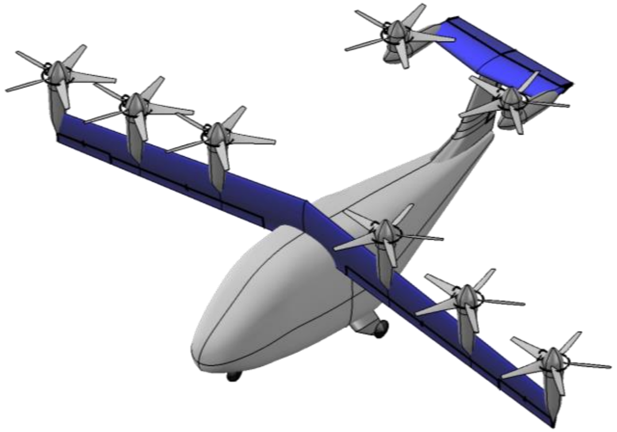
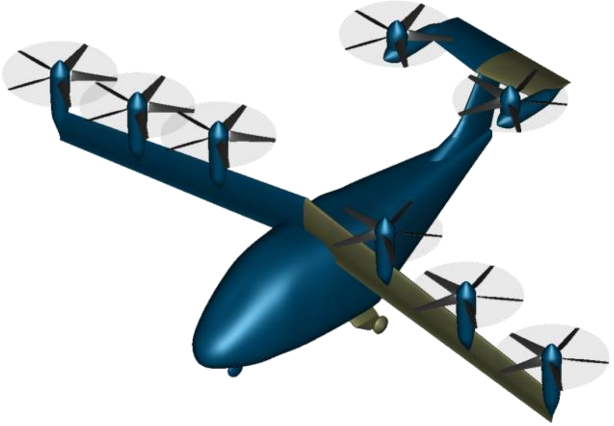
Quite Single Main Rotor

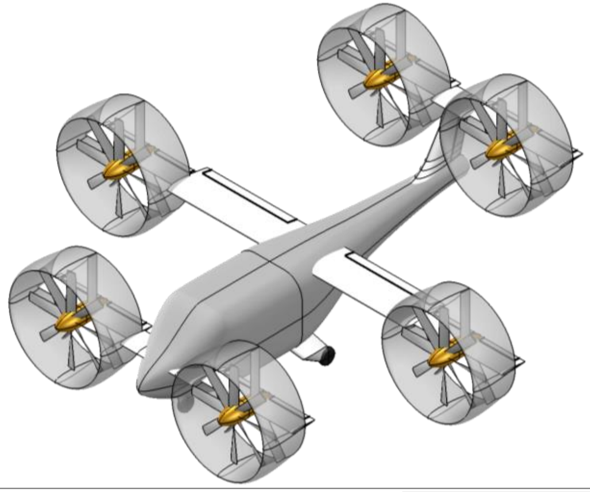
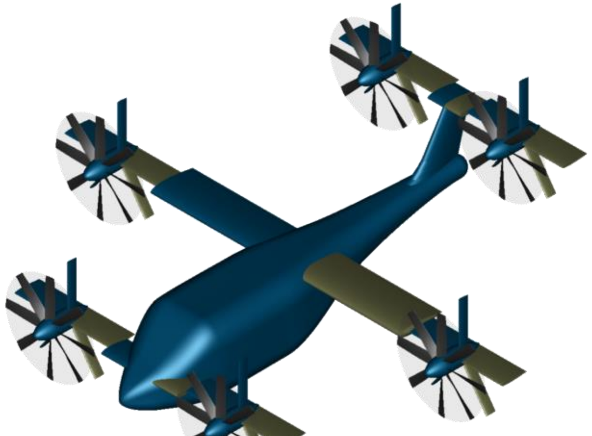
Lift + Cruise

Tiltwing

Tiltduct

No ducts yet!

Models at https://sacd.larc.nasa.gov/uam-refs/

Future work

- Capability improvements
  - Implementation of BOR, Duct and Fuselage with engine settings as <engine>
  - Lofting through guide curves → less sections
  - More detailed control surface parameterization
  - Export of VSPAERO results in CPACS

- Bug fixes
- Public release is planned → integration in OpenVSP or standalone

Links

CPACS

- www.cpacs.de – XSD scheme, documentation and further updates
- github.com/DLR-SL/CPACS_Seminar – Interactive seminar, examples, tool integration

TiXI github.com/DLR-SC/tixi – Code, binary downloads, documentation, issues and updates

TiGL dlr-sc.github.io/tigl/ – Code, binary downloads, documentation, issues and updates

Digital Hangar www.digital-hangar.de (currently fixed wing models only)

DLR rotorcraft research www.dlr.de/en/ft/about-us/departments/department-rotorcraft

THANK YOU FOR YOUR ATTENTION

Kagan Atci
kagan.atci@dlr.de

German Aerospace Center (DLR e.V.)
Institute of Flight Systems | Rotorcraft Department
Lilienthalplatz 7
D-38108 Braunschweig

DLR