20 Years of CPACS
A Hands-On Workshop & Networking Event

DLR

Last updated on: 27 October 2025

Wednesday, 1 October 2025

Pre-Workshop

9:00 – 13:00	**CPACS Design System: How to get started** Pre-Workshop for CPACS newbies *– in-depth hands-on experience for a complete digital engineering process using* CPACS, TiGL and RCE –
13:00 – 14:00	*Lunch Break*

13:00 – 14:00	*Registration & Welcome Coffee*
14:00 – 15:15	**Opening** ▪ Welcome words ▪ CPACS as enabler for Collaborative Digital Engineering (E. Moerland) ▪ 20 years of CPACS: a brief historic recap (C. Liersch)
15:15 –16:00	**Introduction to CPACS Basics** Hands-on tutorials and exercises
16:15 – 17:15	**Visualisation of CPACS Data** An introduction to the interactive DIANA visualisation dashboard
17:15 – 18:00	**CPACS in Practice** ▪ Complex System Architectures in CPACS (T. Burschyk) ▪ Impact Assessment Studies on ATS and Airport Level with CPACS (P. Ratei)
19:30 – 22:00	**Networking Dinner** BLOCKBRÄU – Restaurant at Hamburg Harbour

Thursday, 2 October 2025

8:30 – 9:00	*Registration*
9:00 – 9:45	**Introduction to the TiGL Geometry Library** (J. Kleinert)
9:45 – 10:30	**TiGLCreator: CPACS Geometries from Scratch** Hands-on tutorials and exercises
10:45 – 12:00	**Automation via TiGL Python Scripting** An introduction to the TiGL API with hands-on exercises
12:00 – 13:00	*Lunch Break*
13:00 – 14:00	**TiGL as a Community Project** Interactive discussion with participants
14:00 – 14:45	**The Future of the CPACS Design System** Opportunities to join the community and shape future developments
14:45 – 15:00	**Wrap Up & Closing** (M. Alder)
15:00 – 17:00	**Good-bye Coffee** *CPACS experts will remain available to answer any final questions.*

cpacs A Common Language for Aircraft Design

ZAL TechCenter,
Hamburg, Germany

TiGL

20 YEARS OF CPACS

INTRODUCTION TO THE TIGL GEOMETRY LIBRARY

Hands-On Workshop and Networking Event Hamburg, 2nd October 2025

Sven Goldberg, Jan Kleinert, Meike Kobold

DLR

WHAT IS TIGL?

COMMUNITY AND CONTRIBUTION

CURRENT DEVELOPMENTS

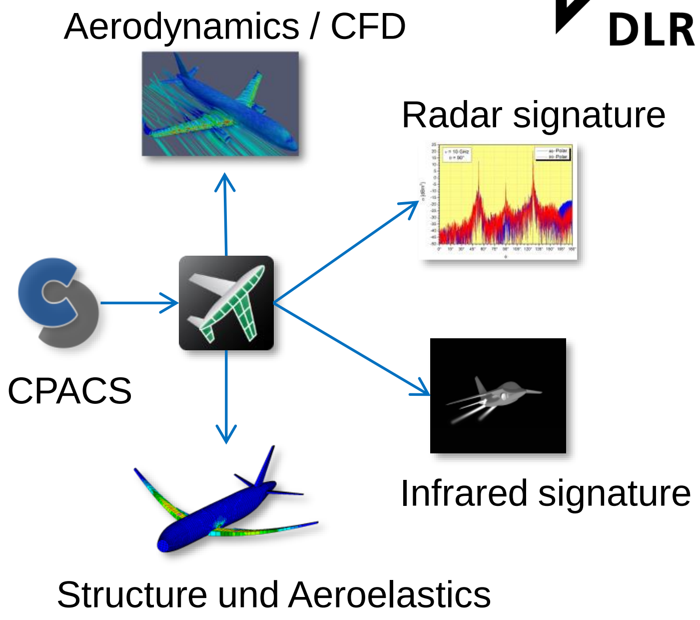
GEOMETRY MODELING: CPACS, TIGL AND BEYOND

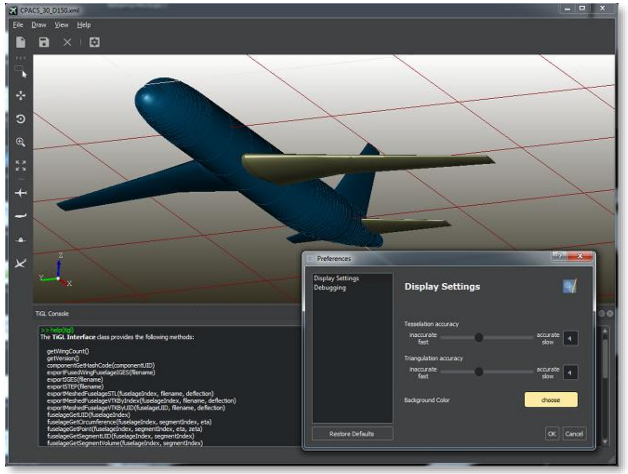
DLR (CC BY 3.0)

WHAT IS TIGL?

The TiGL Software Package

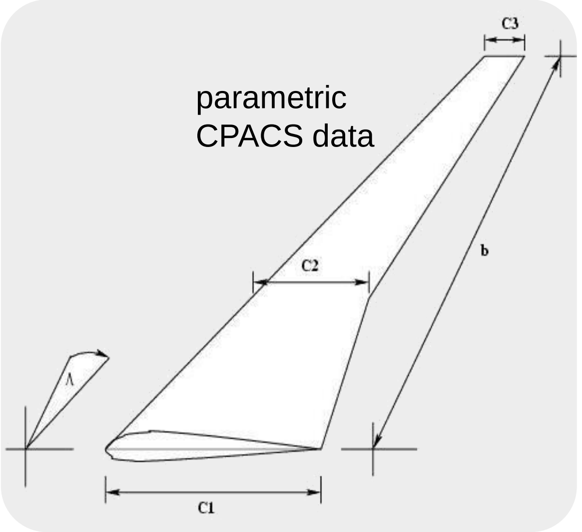
- C++ Library for parametric geometry modelling, design and geometric analysis of aircraft and helicopter
- Based on parametric CPACS 3.4 (XML) files
- TiGLCreator: Visualize and edit CPACS-based aircraft geometries and other CAD files
- Open-Source and cross-platform (Linux, MacOS, Windows)
- Latest Releases:
  - TiGL 3.5.0-rc1 (2025/09/22)
  - TiGL 3.4.1 (2025/09/08)





CPACS – Geometry Description ambiguous

Undefined in CPACS:
- Geometry format
- Tolerances
- Interpolation algorithms
- B-Spline Parameters
- …



All stakeholders still have to use a central library for the geometry generation

Geometry is central component to all involved disciplines

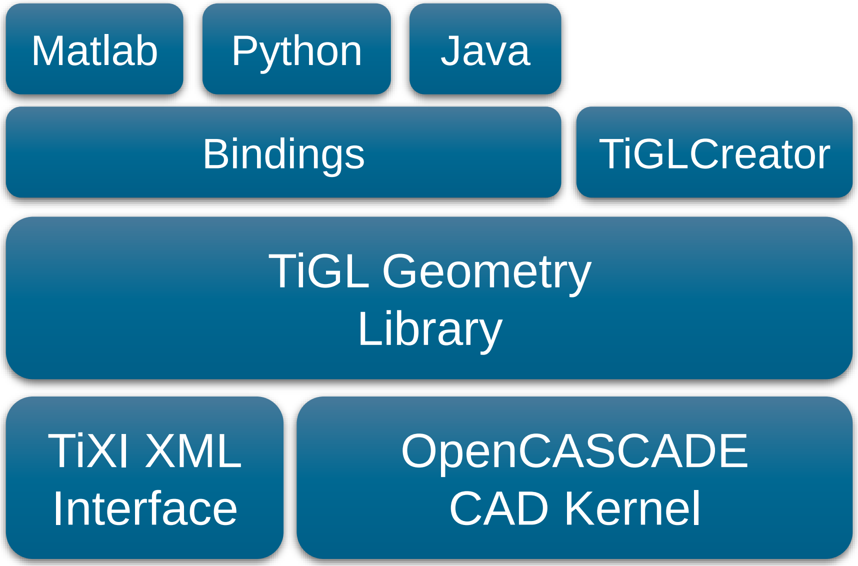
Architecture

- TiXI (https://github.com/dlr-sc/tixi)
  - Library to parse XML (CPACS) files

- OpenCASCADE (https://www.opencascade.com/)
  - Geometry (NURBS-based)
  - Topology (Boundary Representation)
  - CAD Exports, Visualization

- Language Bindings
  - Generated via SWIG (http://www.swig.org/)
  - Can access all C++ Data structures

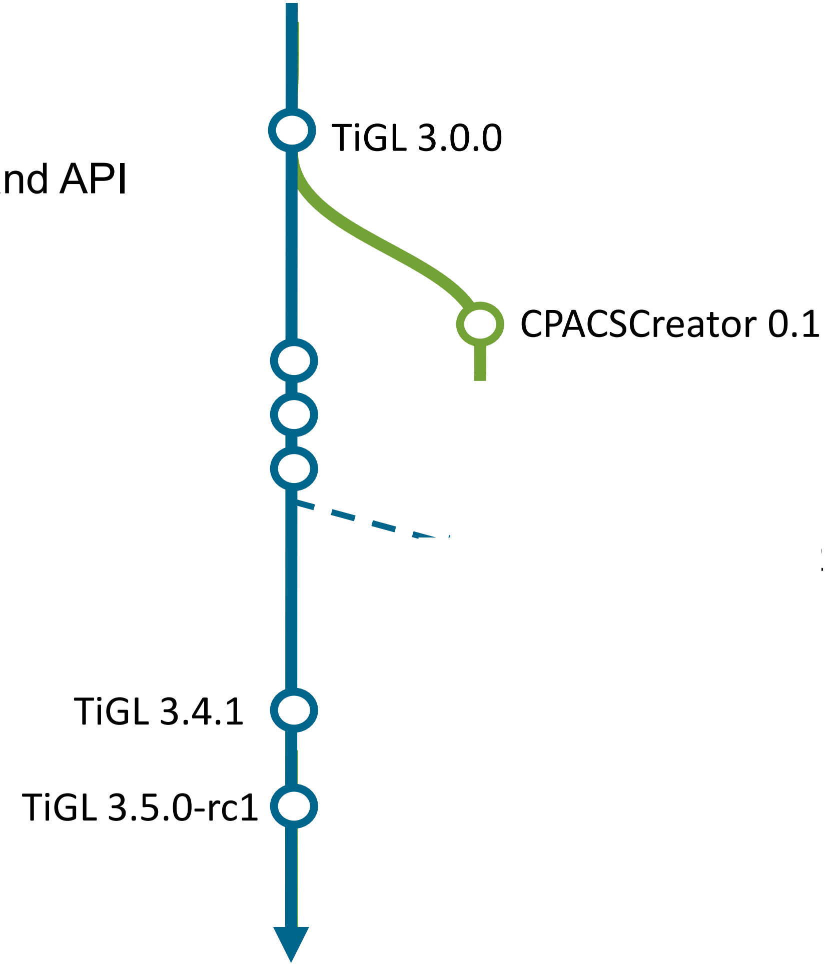
- TiGLCreator
  - 3D Visualization
  - Edit CPACS configurations
  - Scripting, Debugging



TiGLCreator – Historical Overview

- Traditionally TiGLViewer GUI was only a visualization tool
- Idea:
  - Directly edit a CPACS configuration within the GUI
  - Allow editing high-level parameters (sweep, dihedrals, span, …) in GUI and API
- In 2018, Malo Drougard started CPACSCreator in separate branch
  - Finished as part of his Master‘s thesis at CFS Engineering
- First pre-release 0.1 in autumn 2019 (based on TiGL 3.0)

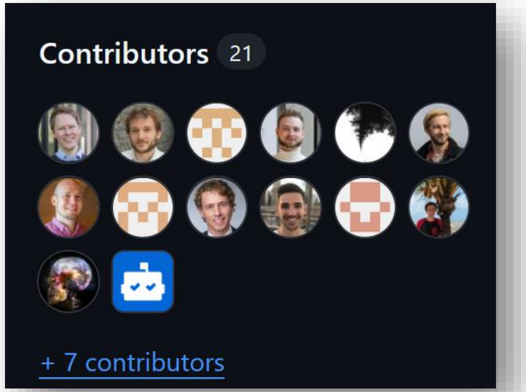
- September 2025:
  - TiGL 3.4.1 last TiGL release without CPACS Creator
  - CPACS Creator merged into TiGL main branch
  - TiGL 3.5.0-rc1 **as early preview for this Workshop**



TiGLViewer/CPACSCreator GUI renamed to **TiGLCreator**

TiGL Team

- TiGL is an important tool for us and we keep support and development up
- Current main developers: 4 + 1 student (all part time)
- Many contributions from DLR AIRBUS DEFENCE & SPACE RISC Software GmbH
- and others




Ole Albers


Sven Goldberg


Jan Kleinert


Meike Kobold


Anton Reiswich

COMMUNITY AND CONTRIBUTION

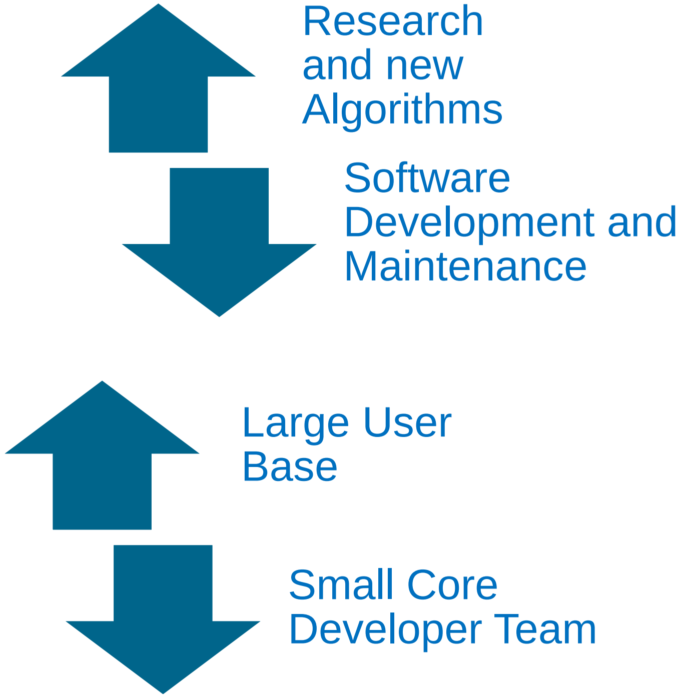
TiGL as a community-developed Tool

- TiGL is and will stay free and open-source
- Widely used in research at the DLR, international universities and in industry

- Software Maintenance takes up time for Research
  - Bug fixes, Refactoring
  - Continuous Integration, Updates and Releases
  - Documentation, Q&A, Support, …

- Challenge: High entrance barrier for contributions
  - C++, Qt, OCCT, …
  - TiGL, TiXI, CPACS, ….

→ **User contribution is needed**



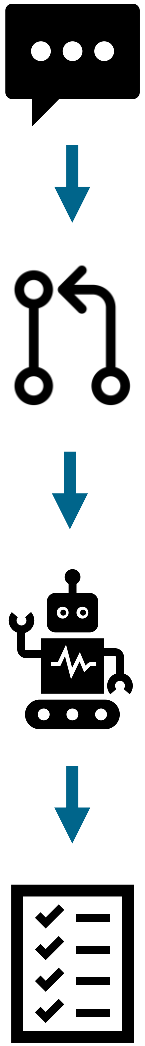
Different Ways to Contribute

- **Bug reports** (create an **issue** on GitHub)
- New **examples** (CPACS definitions, geometric definitions in PythonOCC or OCCT)
- Add or improve **documentation,** or **website**
- **Code** (Bug fixes or implementation of new features)

  → Workflow: **Issue** and **pull request**

You can also write us an email or give us a call with improvement suggestions or comments!

Code Contribution Workflow



1. **Create Issue**
   - Describe need for new feature or bug report

2. **Create Pull Request**
   - Refer to existing issue *OR* create new one
   - Rule of thumb:
     - Every new functionality should come with a new test
     - Code coverage should not decrease

3. **Automated CI pipeline** must succeed
     - CI configures and builds on different OS
     - executes unit- and integration tests
     - checks code coverage *(ratio of tested lines of code)*

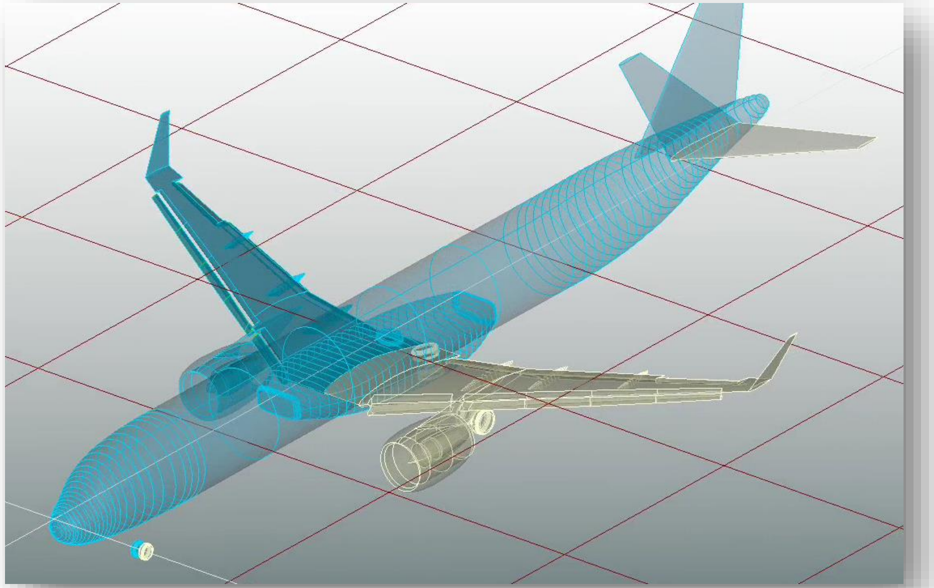
4. **Code review** by a TiGL main developer is required before merge

CURRENT DEVELOPMENTS

Current developments in TiGL

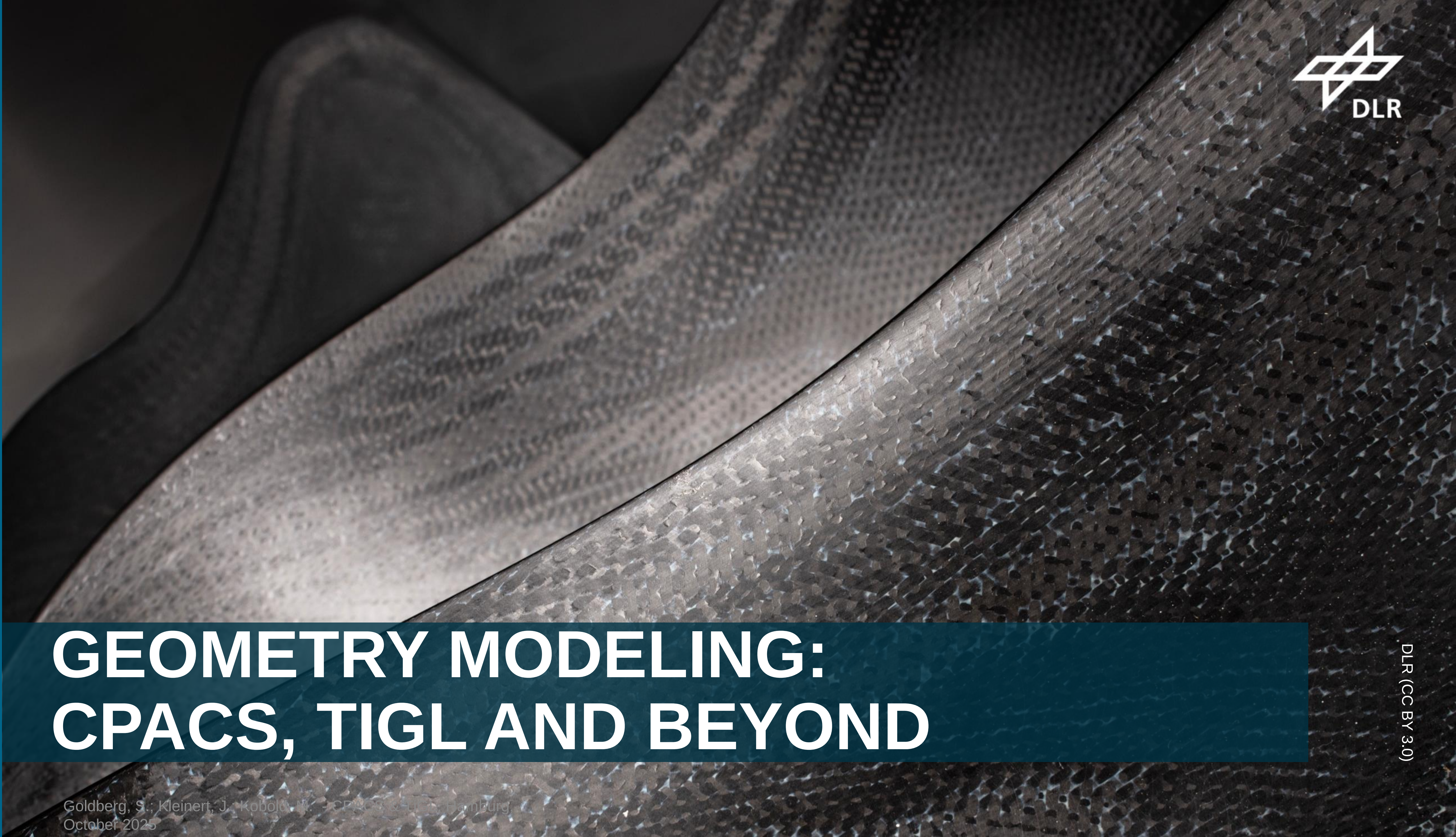
Ongoing work:

- **Stability**: Fix bugs and test new functionality from the CPACS Creator merge
- **Surface Quality**: New low-level algorithms for smooth lofting of wings with rounded sections
- Support **Leading Edge Devices** and Spoilers
- **TiGL Creator**: Persistent Session (camera parameters, visible geometric components), fine control over visualization and geometry export



Community needs:

- Lets discuss this afternoon!



GEOMETRY MODELING: CPACS, TIGL AND BEYOND

DLR (CC BY 3.0)

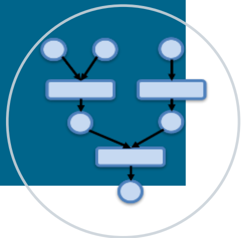
Research interests in Geometry Modeling

- Geometry representation (B-Splines, NURBS, …)
- Interpolation and Approximation
- Blending curves/surfaces
- Boolean operations
- Geometry reconstruction
- …

CAD Algorithms

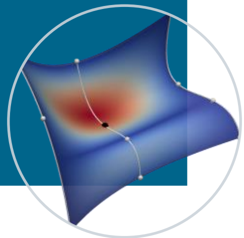
- Topological/Persistent naming
- Data Flow Modeling
- Functional Programming
- Generic Programming
- AI Assistant
- …

Parametric Design



- Algorithmic Differentiation
- Shape Matching and Reconstruction
- Adjoint Simulation?
- …

Shape Optimization



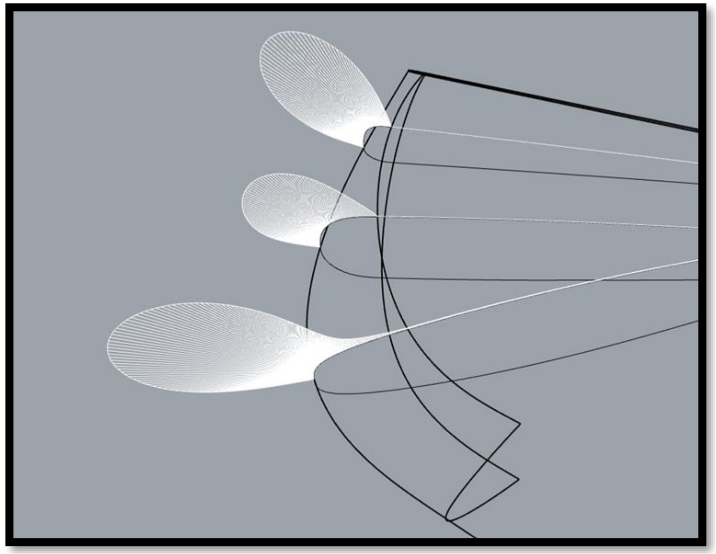
Software

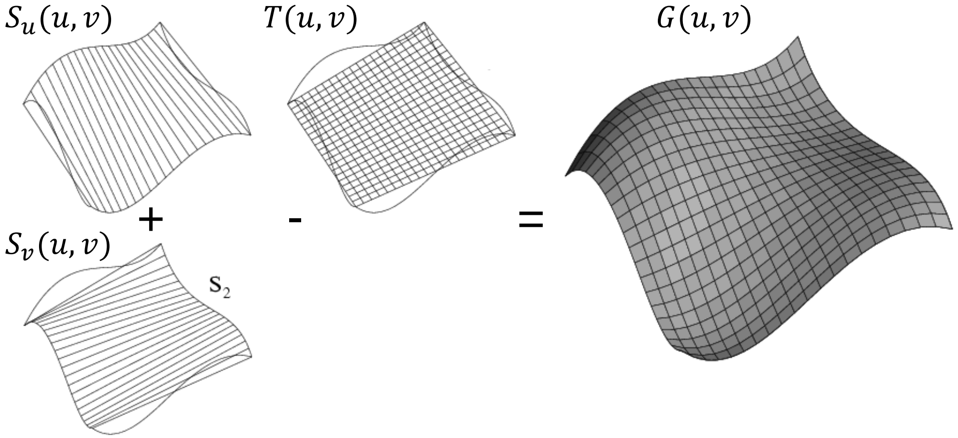
- TiGL, geoml, grunk

geoml – Geometry Modeling library

- TiGL Spin-Off for generic geometry modeling
- C++ library with python bindings
- Free and opensource
- Based on OpenCASCADE TECHNOLOGY

- Well-structured tool box for geometry modeling using **NURBS/B-Splines**
- Advanced functions such as **Gordon surfaces**
- Further extensions of OCCT
- Persistent naming and generic **meta data** system (WIP)





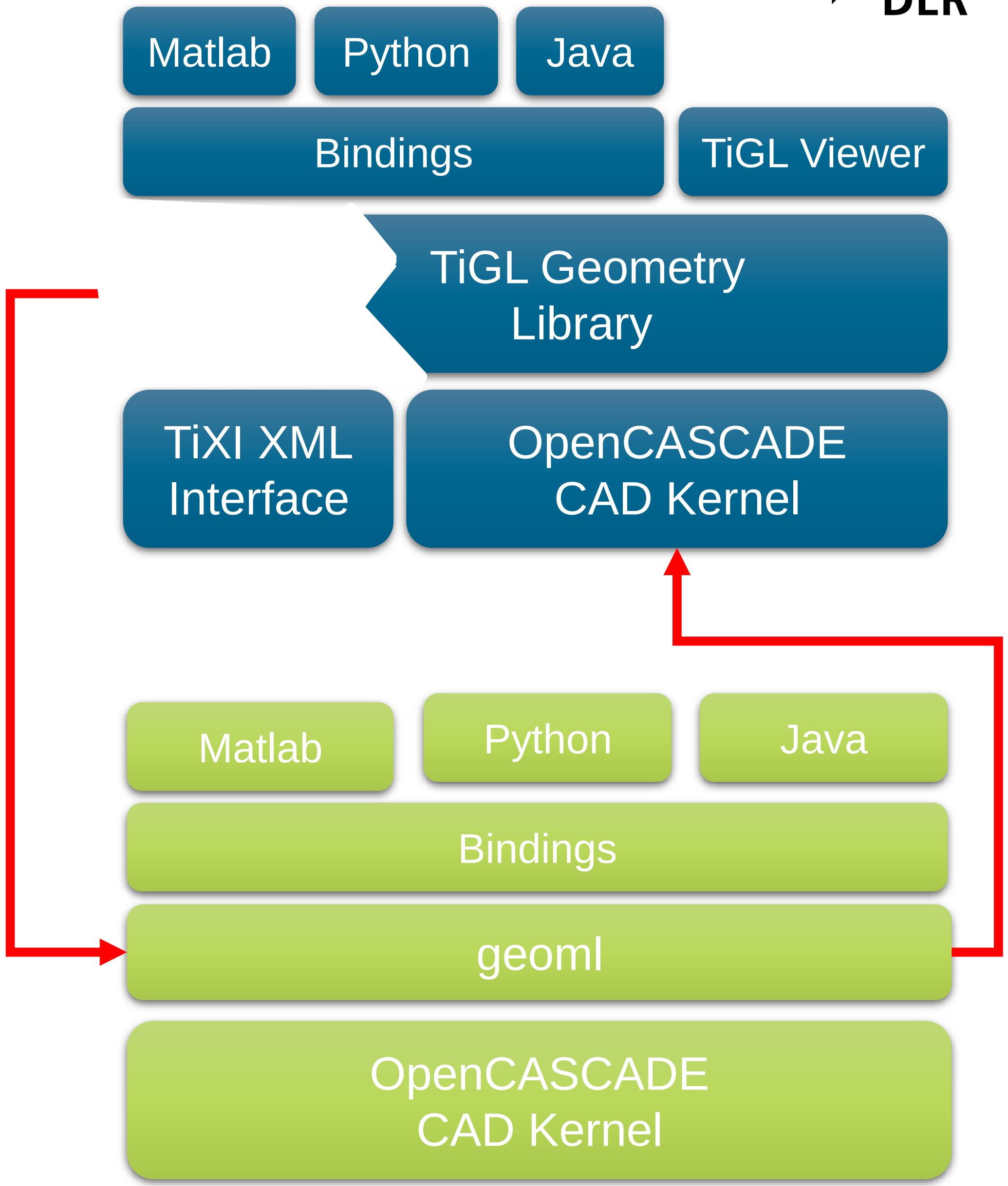
geoml

- Done:
  - Extracted geoml as a stand-alone library from TiGL
  - Publish geoml as FOSS on Github

- Next steps:
  - Link TiGL against geoml
  - Remove geoml src from TiGL repo

Seperation of concerns:

- TiGL for CPACS manipulation
- Geoml for geometry algorithms



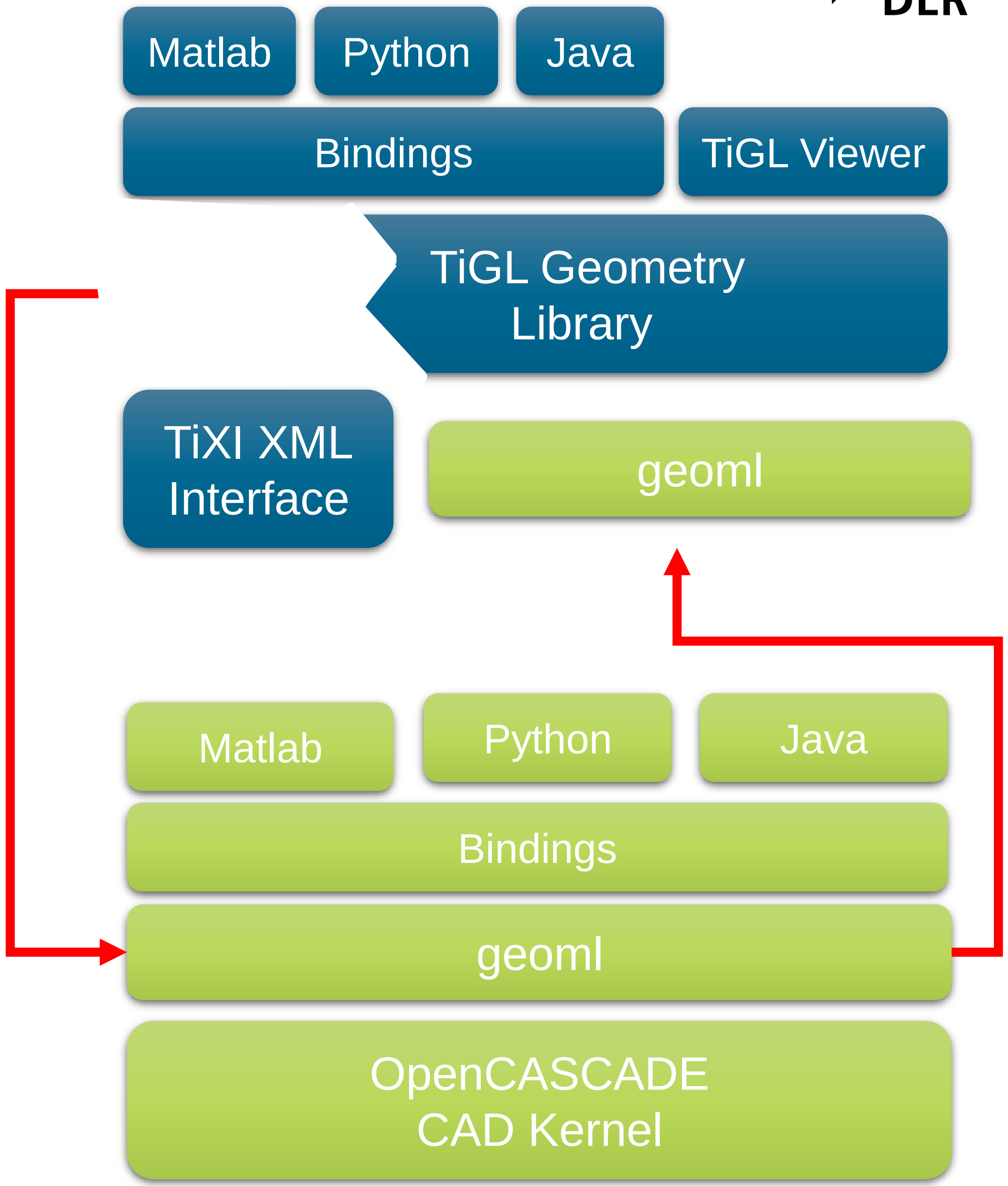
geoml

- Done:
  - Extracted geoml as a stand-alone library from TiGL
  - Publish geoml as FOSS on Github

- Next steps:
  - Link TiGL against geoml
  - Remove geoml src from TiGL repo

Seperation of concerns:

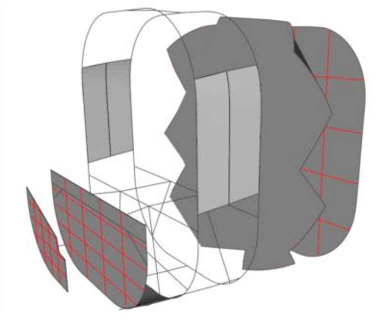
- TiGL for CPACS manipulation
- Geoml for geometry algorithms

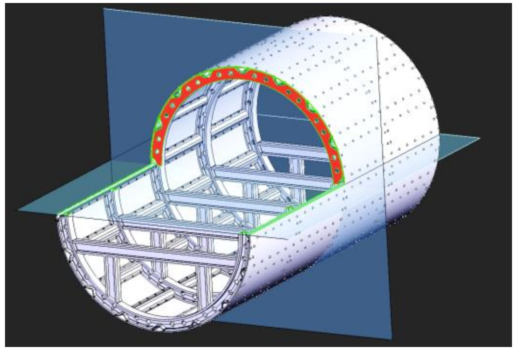


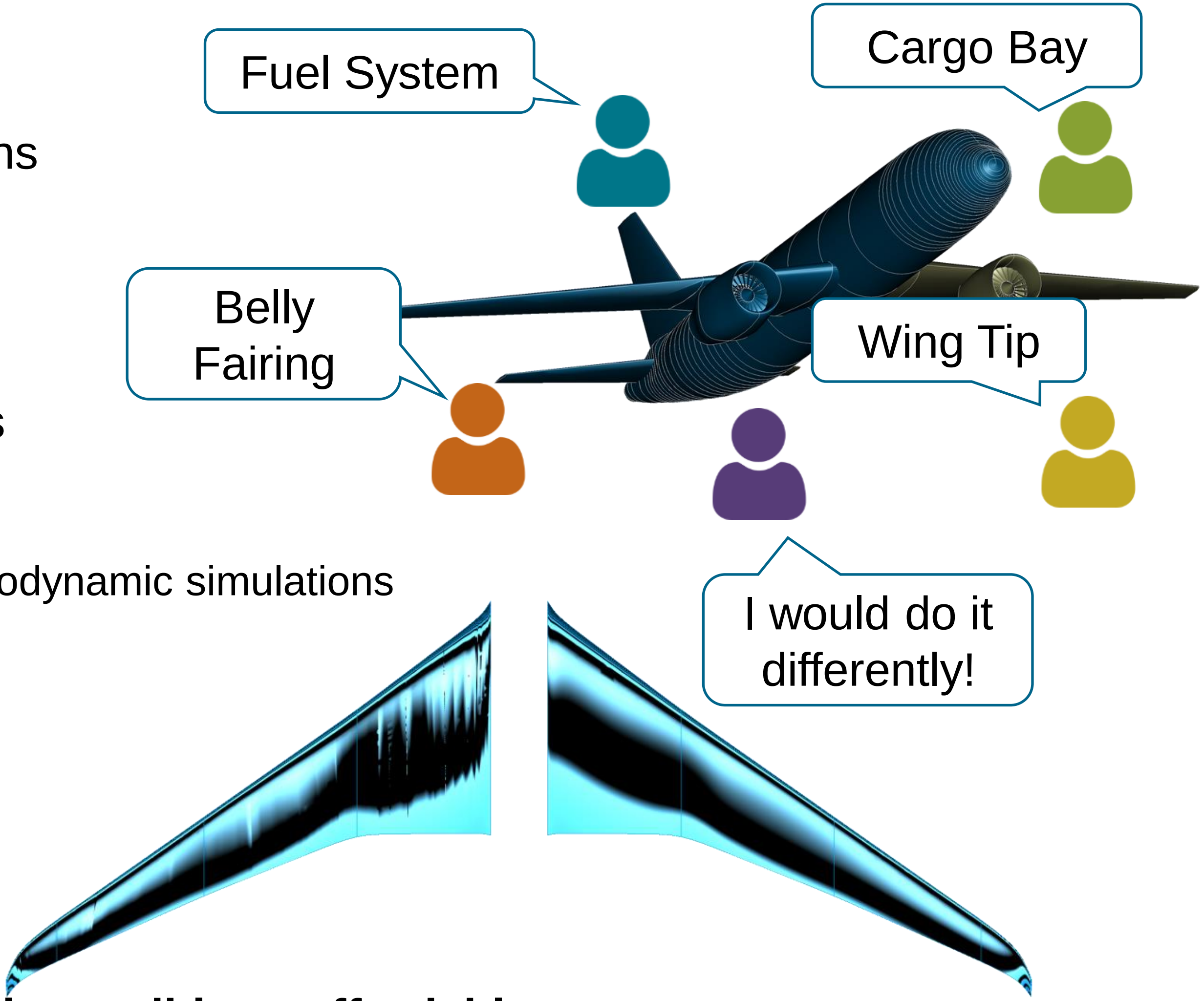
Cutting Edge Research needs Custom Solutions

The Challenge

- CPACS might be insufficient for unconventional designs
  - Missing features
  - Different parametrization needed
- TiGL geometry might be inadequate for specific needs
  - “HiFi” vs “LoFi”
  - Different surface quality criteria for e.g. structural or aerodynamic simulations





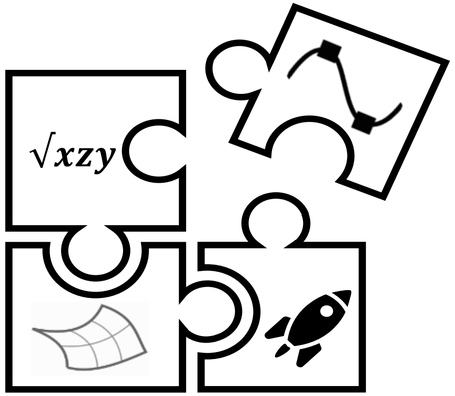


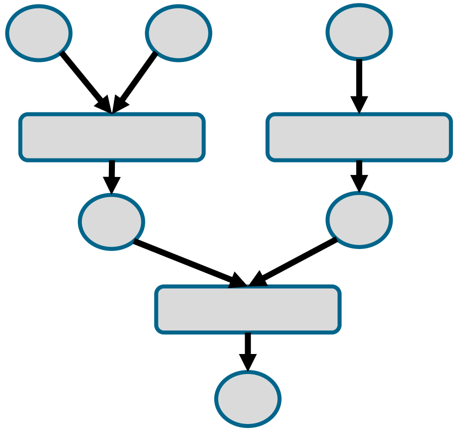
→ Extension of CPACS/TiGL not always meaningful, possible or affordable

Solution: Parametric engine grunk

- **Modeling driver for heterogenous tools via plugin system**
  from different departments and disciplines
  - CAD (OpenCascade, geoml, …)
  - KBE (Codex, FUGA, …)
  - CPACS and other data models (TiGL, Gtlab, …)

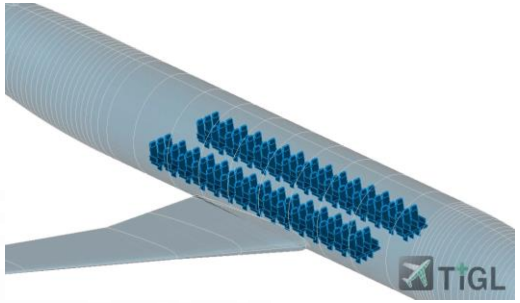
- Hierarchical human-readable **exchange format** based on yaml
- Extendable **single source of truth** across departments
- **Multifidelity** support by design





<u>Before</u>

- CPACS as data model („standard“)
- TiGL closes ambiguity in geometric interpretation of CPACS

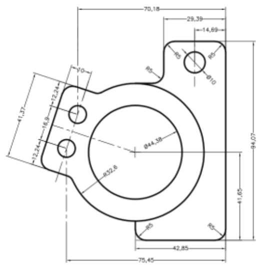


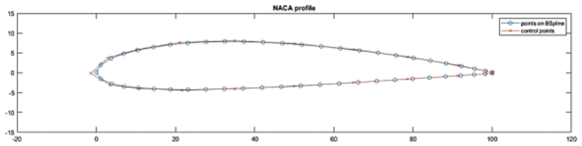
<u>Now additionally</u>

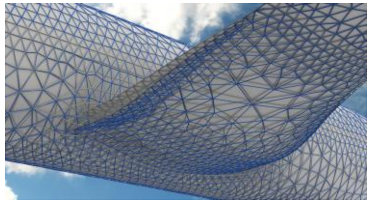
- Extend CPACS with custom solutions that go beyond the standard
- Share extended model across departments and disciplines

Goal: Shape Optimization – for ALL disciplines

All disciplines | **Single discipline**







Parametric Design → B-Spline representation → Meshing → Simulation (adjoint CFD) → Cost Function (e.g. fuel consumption)

Gradient-based optimization

Unique Expertise at DLR | **State-of-the-art**

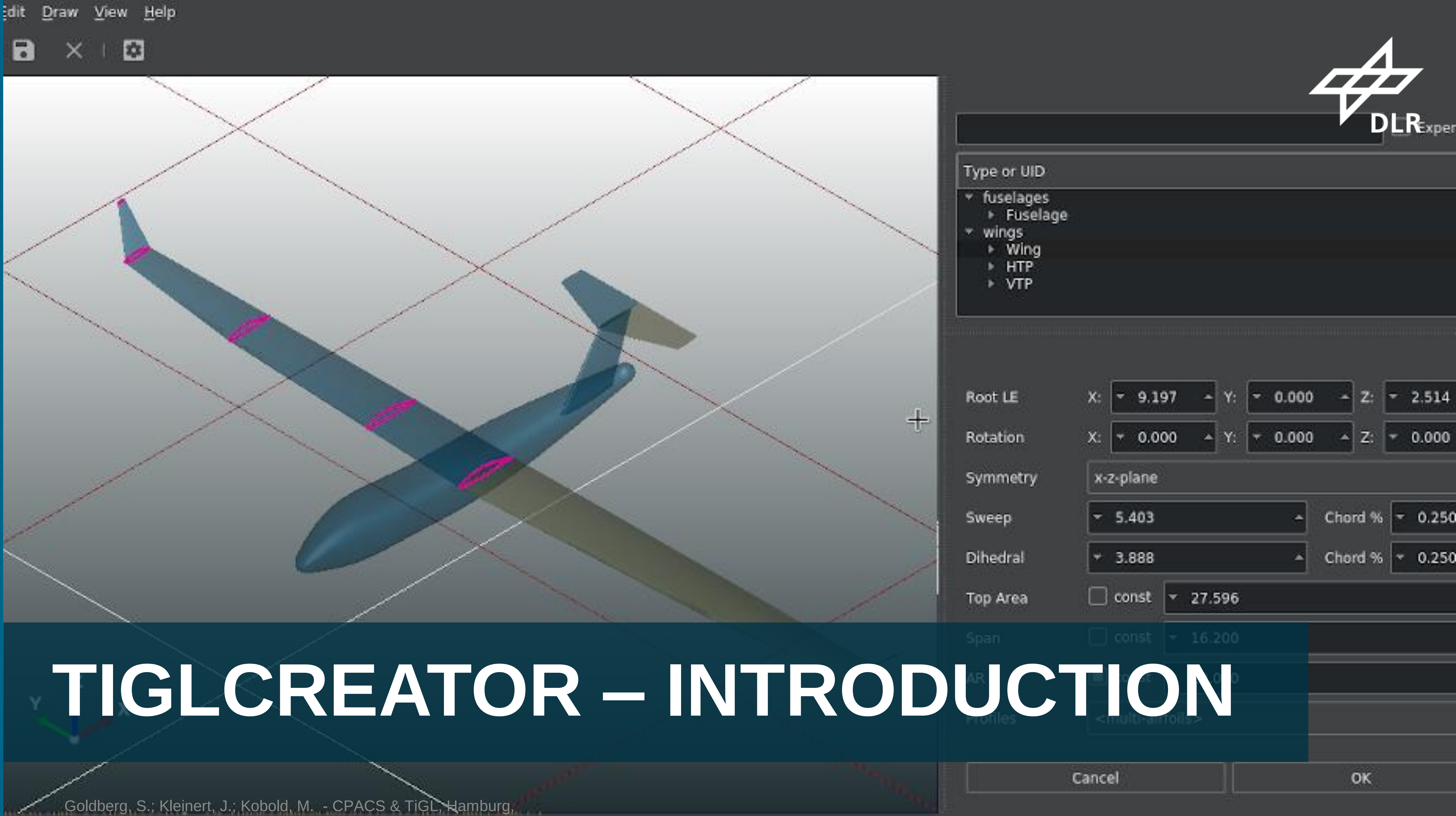
https://commons.wikimedia.org/wiki/File:AcotacionTecnico.svg

20 YEARS OF CPACS

TIGLCREATOR: CPACS GEOMETRIES FROM SCRATCH

Hands-On Workshop and Networking Event Hamburg, 2nd October 2025

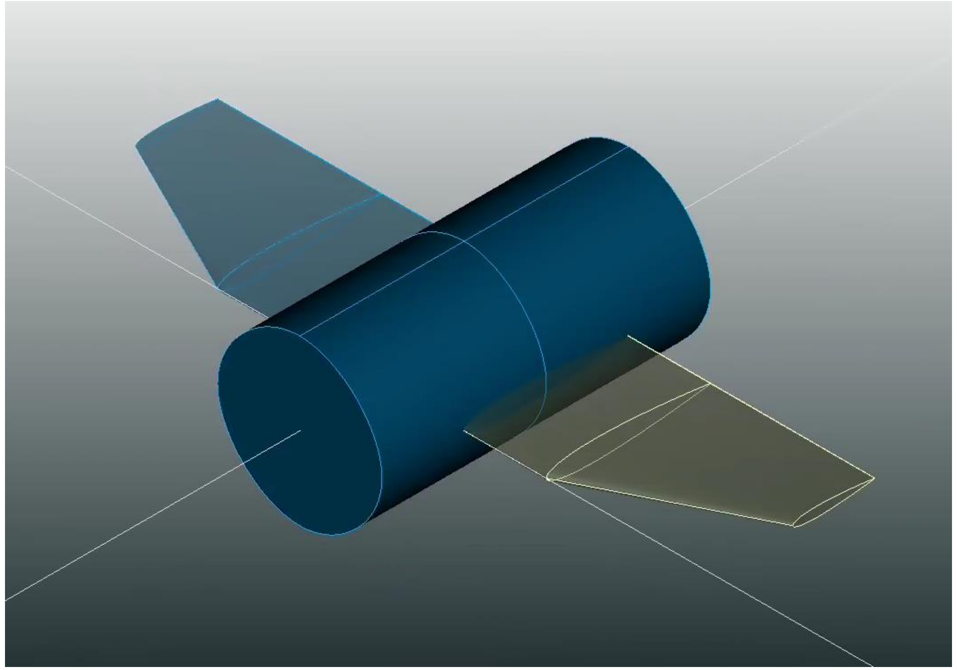
Sven Goldberg, Jan Kleinert, Meike Kobold

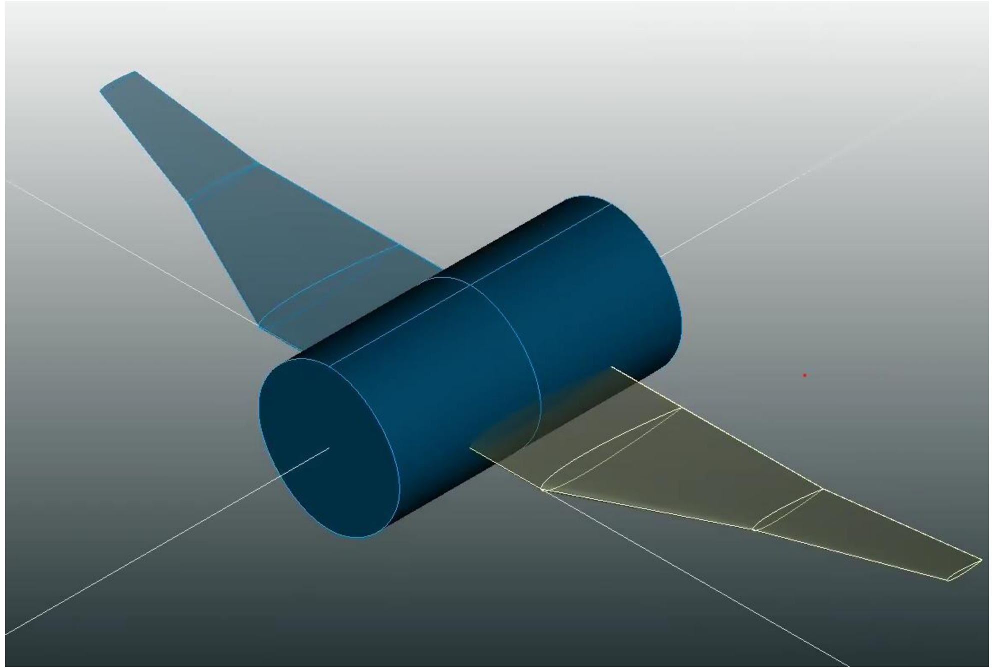


TIGLCREATOR – INTRODUCTION

Changing a Plane‘s Shape

- What if I want to change a plane‘s shape?
  - … e.g. enlarge the wing and change its form by adding another section





Changing a Plane‘s Shape – XML Style

1. Manually add section in CPACS file
   - Think of meaningful parameters

```xml
<section uID="Cpacs2Test_Wing_Sec3">
  <name>Cpacs2Test - Wing Section 3</name>
  <description>Cpacs2Test - Wing Section 3</description>
  <transformation uID="Cpacs2Test_Wing_Sec3_transformation1">
    <scaling uID="Cpacs2Test_Wing_Sec3_transformation1_scaling1">
      <x>1</x>
      <y>1</y>
      <z>1</z>
    </scaling>
    <rotation uID="Cpacs2Test_Wing_Sec3_transformation1_rotation1">
      <x>0</x>
      <y>0</y>
      <z>0</z>
    </rotation>
    <translation refType="absLocal" uID="Cpacs2Test_Wing_Sec3_transformation1_translation1">
      <x>0</x>
      <y>0</y>
      <z>0</z>
    </translation>
  </transformation>
  <elements>
    <element uID="Cpacs2Test_Wing_Sec3_El1">
      <name>Cpacs2Test - Wing Section 3 Main Element</name>
      <description>Cpacs2Test - Wing Section 3 Main Element</description>
      <airfoilUID>NACA0012</airfoilUID>
      <transformation uID="Cpacs2Test_Wing_Sec3_El1_transformation1">
        <scaling uID="Cpacs2Test_Wing_Sec3_El1_transformation1_scaling1">
          <x>0.5</x>
          <y>0.5</y>
          <z>0.5</z>
        </scaling>
        <rotation uID="Cpacs2Test_Wing_Sec3_El1_transformation1_rotation1">
          <x>0</x>
          <y>0</y>
          <z>0</z>
        </rotation>
        <translation refType="absLocal" uID="Cpacs2Test_Wing_Sec3_El1_transformation1_translation1">
          <x>0.5</x>
          <y>0</y>
          <z>0</z>
        </translation>
      </transformation>
    </element>
  </elements>
</section>
```

2. Manually add segment in CPACS file

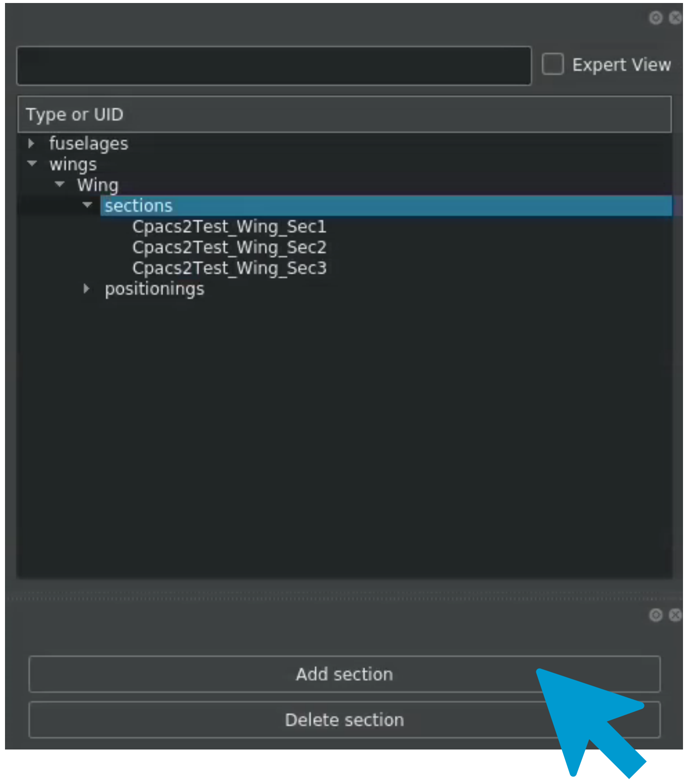
```xml
<segments>
  &lt;segment uID="Cpacs2Test_Wing_Seg_1_2"&gt;
    <name>Fuselage Segment from Cpacs2Test - Wing Section 1 Main Element to Cpacs2Test - Wing Section 2 Main Element</name>
    <description>Fuselage Segment from Cpacs2Test - Wing Section 1 Main Element to Cpacs2Test - Wing Section 2 Main Element</description>
    <fromElementUID>Cpacs2Test_Wing_Sec1_El1</fromElementUID>
    <toElementUID>Cpacs2Test_Wing_Sec2_El1</toElementUID>
  &lt;/segment&gt;
  &lt;segment uID="Cpacs2Test_Wing_Seg_2_3"&gt;
    <name>Fuselage Segment from Cpacs2Test - Wing Section 2 Main Element to Cpacs2Test - Wing Section 3 Main Element</name>
    <description>Fuselage Segment from Cpacs2Test - Wing Section 2 Main Element to Cpacs2Test - Wing Section 3 Main Element</description>
    <fromElementUID>Cpacs2Test_Wing_Sec2_El1</fromElementUID>
    <toElementUID>Cpacs2Test_Wing_Sec3_El1</toElementUID>
  &lt;/segment&gt;
</segments>
```

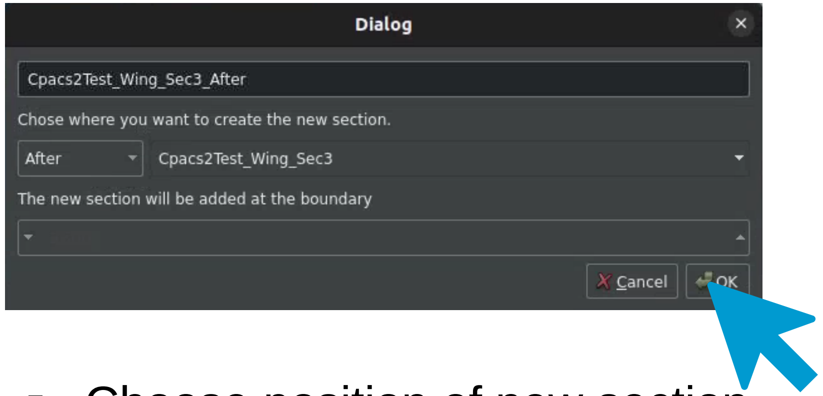
3. Some more smaller steps…

→ Much user interaction and several manual changes of the CPACS file are needed

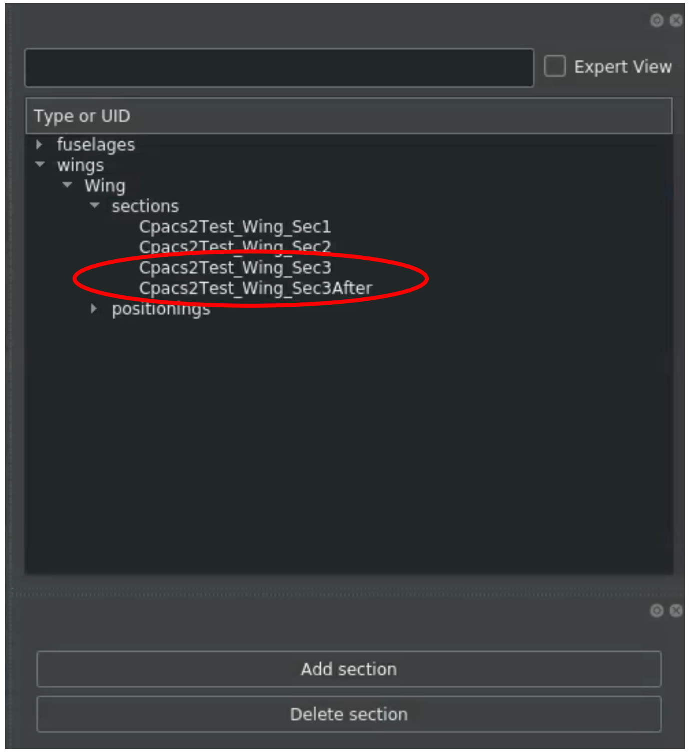
Changing a Plane‘s Shape – TiGLCreator Style

- Back to example: Add a section



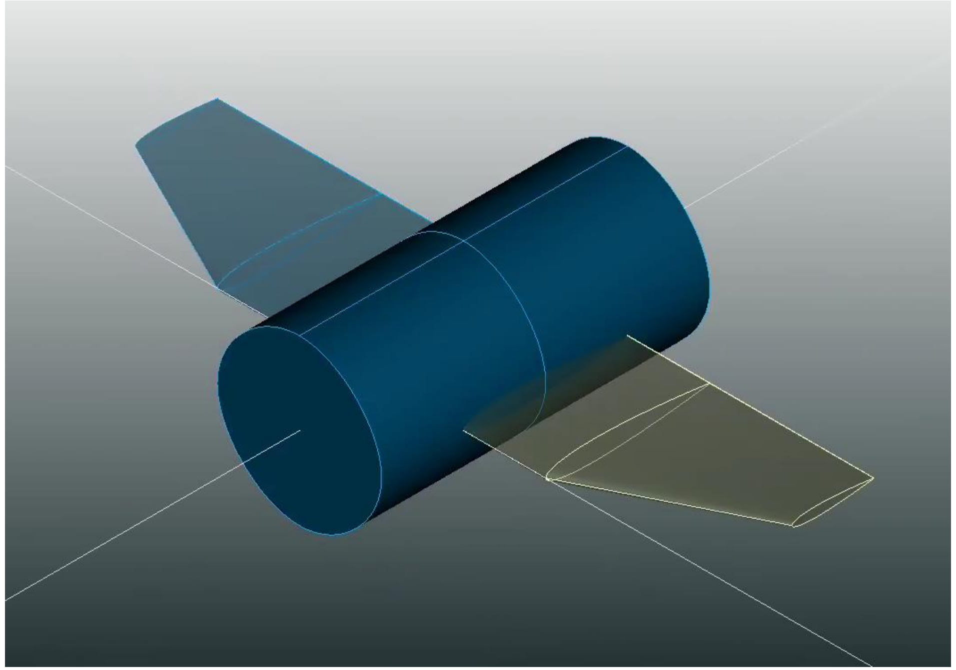


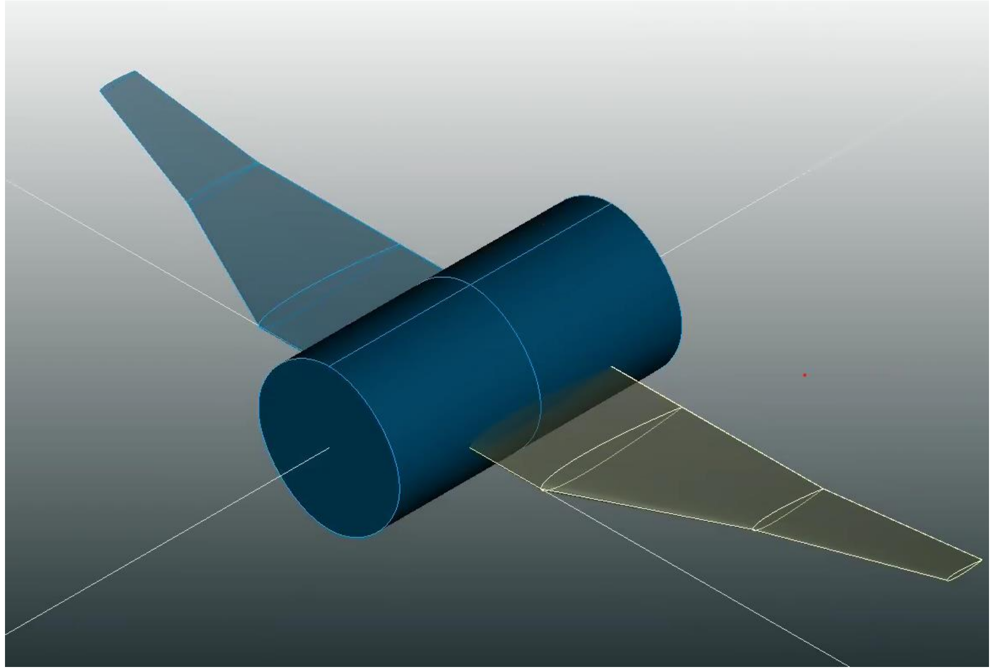
- Choose position of new section
- Choose name (or use default)



TiGLCreator – Example

- Back to example: Add a section





- Many more features …

TiGLCreator – Example

DEMO

TiGLCreator – Features

Fuselages and Wings

- Add, edit and delete sections/segments
- High-level parameter directly editable
  - Manual change in CPACS file is elaborately
- Change the wing and fuselage positionings (→ total length, e.g.)

Wings

- Change the wing‘s symmetry

General

- Represent CPACS file in tree view
- Undo / redo buttons
- Save edited CPACS file

TiGLCreator – Limitations

Limitations (up to now …)

- Only wings and fuselages
- Not possible to add sections in segments with guide curves
- Only one fuselage profile
- Only small list of airfoils
- New profiles must be added to CPACS file manually
- Low-level parameter not directly adjustable (transformation, e.g.)

TIGLCREATOR – HANDS-ON

TiGLCreator – Hands-On

- Download the TiGLCreator executable (Link sent via mail)

- Open the file TiGL/data/swift.cpacs.xml in the workshop folder
  - Folder: https://jhub.mbse-env.com/ (Username and PW from yesterday)
  - CAD file in .stp-format is shown
  - CPACS Configuration does not contain any fuselages or wings

- Idea: Recreate this plane from scratch using the TiGLCreator

- Familiarize yourself with the TiGLCreator, no perfect rebuild

TiGLCreator – Hands-On: Hints

- **Fuselage**:
  - Start by adding a new fuselage containing 8 sections
  - Change high-level fuselage parameters (length, width, height) first
  - Then, move on to more detailed section parameters
    (section center -> section width/height)

- **Wings**:
  - For all wings, 2 sections are enough for this tuturial
  - Start with the main wing or HTP (design only one due to symmetry)
  - Move the sections‘ center to build the wing
  - Again, choose meaningful values for area, width and height
  - Finally, add the VTP. Hint: Rotate the whole wing

20 YEARS OF CPACS

AUTOMATION VIA TIGL PYTHON SCRIPTING

Hands-On Workshop and Networking Event Hamburg, 2nd October 2025

Sven Goldberg, Jan Kleinert, Meike Kobold

Python bindings come in two flavors

High-Level

- Since 2015
- Exposes TiGL C API
- Simple: Hides internal complexity
- No direct interaction with geometry

```python
def main():
    tixi_handle = tixi3wrapper.Tixi3()
    tigl_handle = tigl3wrapper.Tigl3()

    dir_path = os.path.dirname(os.path.realpath(__file__))
    tixi_handle.open(dir_path + "/../../tests/unittests/TestData/simple
    tigl_handle.open(tixi_handle, "")

    px, py, pz = tigl_handle.wingGetChordPoint(1, 1, 0.5, 0.5)

    ...
```

Low Level / „internal“

- Since 2018, uses SWIG bindings generator
- Exposes (almost) all C++ classes
- Highly customizable, full interoperability with pythonOCC
- Not well documented

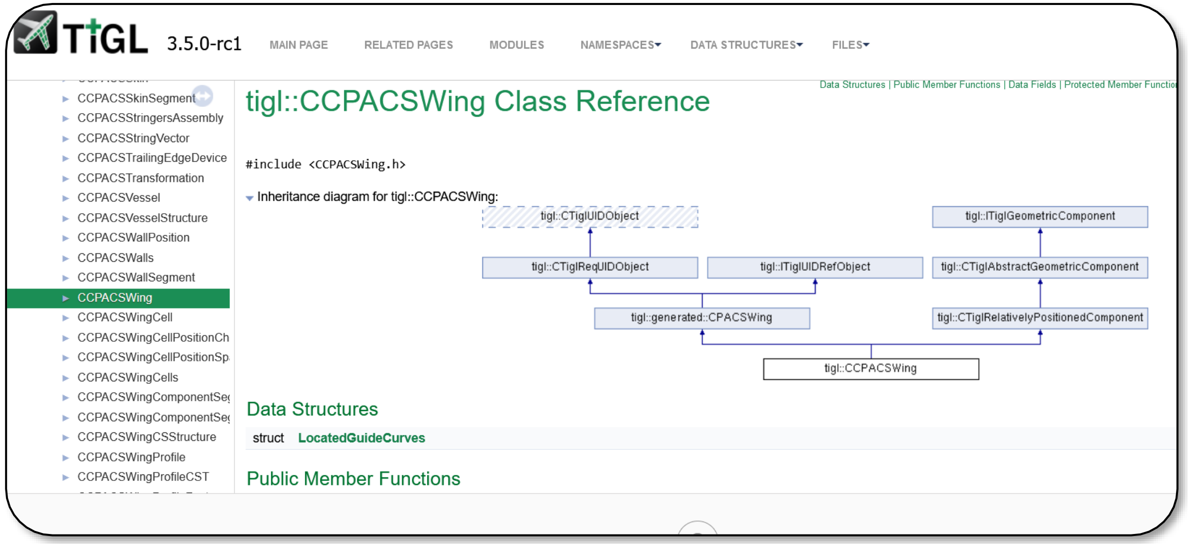
```python
        xsi4 = leading_edge_border.get_xsi_1_choice2()

        bb = Bnd_Box()
        brepbndlib.Add(shell_shape, bb)
        Xmin, Ymin, Zmin, Xmax, Ymax, Zmax = bb.Get()

        pnt1 = self._component_segment.get_point(eta1, xsi1)
        pnt2 = self._component_segment.get_point(eta2, xsi2)
        pnt3 = self._component_segment.get_point(eta3, xsi3)
        pnt4 = self._component_segment.get_point(eta4, xsi4)
        pnt1.SetZ(Zmin)
        pnt2.SetZ(Zmin)
```

Design Philosophy

- Direct unaltered access to the internal C++ API
- Full **interoperability** with pythonOCC
- Provide optional wrapper functions to make use **more pythonic**
- Use **C++ documentation** of the class hierarchy and translate directly to Python
  (e.g. CamelCase to snake_case etc.)

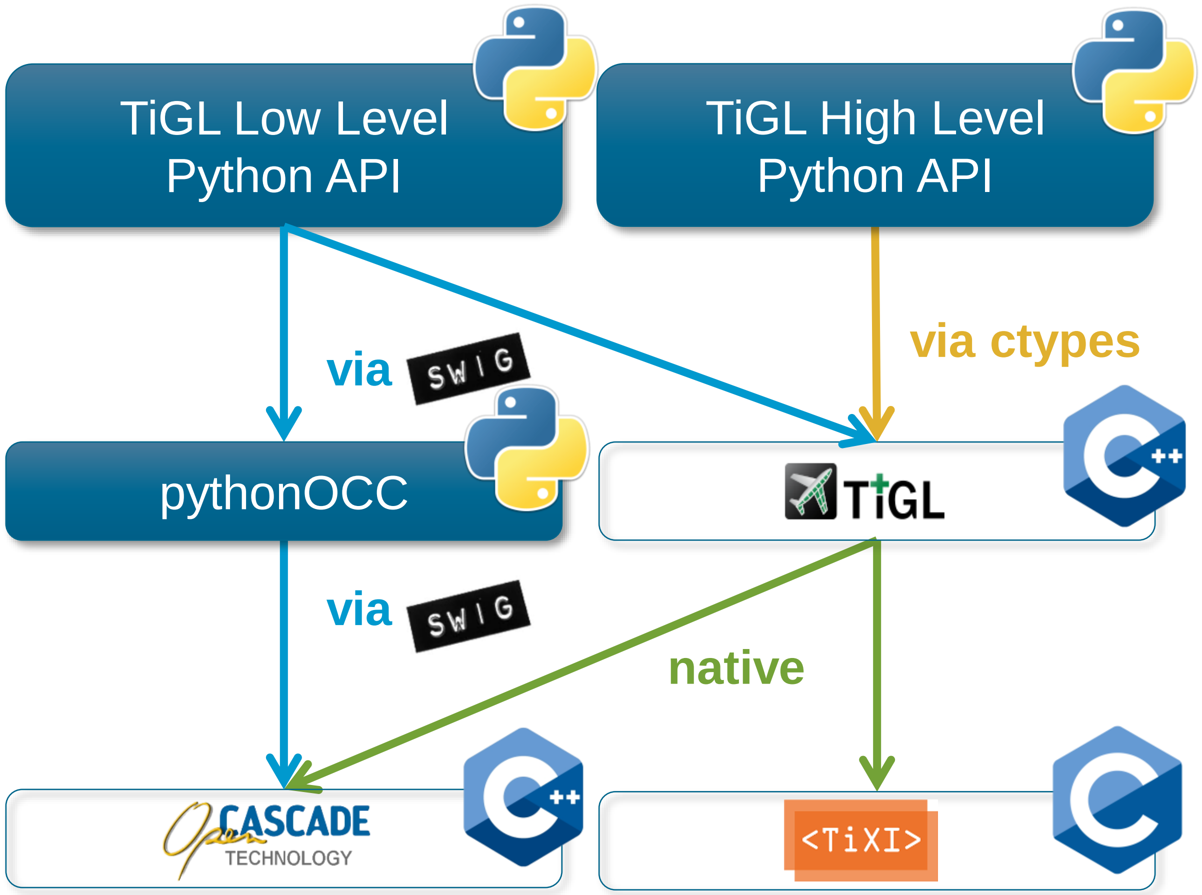


Architecture

- OpenCASCADE bindings **pythonOCC** generated with SWIG software: http://www.swig.org/

- TiGL low-level API also generated with SWIG from TiGL & pythonOCC

- Objects between pythonOCC and TiGL can be exchanged



TIGL Low Level Python API: Module overview

DLR

Module	Description
`tigl3.configuration`	Access to the whole CPACS tree including wings, fuselages, …
`tigl3.geometry`	Functions related to geometry operations, e.g. curve interpolation
`tigl3.curve_factories`	Function to create curves based on `tigl3.geometry`
`tigl3.surface_factories`	Functions to create surfaces based on `tigl3.geometry`
`tigl3.occ_helpers`	Pythonic wrappers to OCCT functions
`tigl3.exports`	Classes and functions for file export
`tigl3.imports`	Classes and functions for file import
`tigl3.boolean_ops`	Classes and functions to perform Boolean operations
`tigl3.tmath`	A few math functions
`tigl3.core`	Core functionality, like error handling

Configuration Manager

- TiGL handles several aircraft configurations at a time. To get a specific one, use the **configuration manager class:**

```
from tigl3 import tigl3wrapper
from tigl3.configuration import CCPACSConfigurationManager_get_instance

tigl_handle = tigl3wrapper.Tigl3()
tigl_handle.open(tixi_handle, "")

mgr = CCPACSConfigurationManager_get_instance()
aircraft_config = mgr.get_configuration(tigl_handle._handle.value)
```

Load the high-level Python API

Retrieve the global `CCPACSConfigurationManager` class

Access a specific aircraft given its tigl handle (an integer id)

- `aircraft_config` is instance of the class `CCPACSConfiguration`. Gives direct access to wings, fuselages, systems, …

CPACS Traversal from a CCPACSConfiguration object

- Use the python internal help function to print methods and members of each CPACS node

```
help(aircraft_config)
```

- Example usage:

```
wing = aircraft_config.get_wing(1)
wing = aircraft_config.get_wing("wing_uid")
wing.get_uid()

segment = wing.get_segment(2)
segment = wing.get_segment("segment_uid")

aircraft_config.get_fuselage(1)
aircraft_config.get_uid()

aircraft_config.get_wing(1).get_component_segment(1).get_structure().get_spar_segment(1)
    .get_spar_cross_section().get_rotation()
```

The UID Manager

- Most objects that have a CPACS uid are registered at the **UID manager** when reading the CPACS file

- The UID Manager is **a member of the `CCPACSConfiguration`** object
- The UID Manager enables **direct access** to those objects:

```
uid_mgr = aircraft_config.get_uidmanager()
wing  = uid_mgr.get_geometric_component("WingUID")
spars = uid_mgr.get_geometric_component("SparSegmentsUID")
```

CNamedShape

- Anything derived from `CTiGLAbstractGeometricComponent` has a `get_loft` method
- `get_loft` returns a `CNamedShape` instance
- CNamedShape is a wrapper around an OCCT `TopoDS_Shape` with additional metadata (used in Boolean ops and exports

```
shape = wing.get_loft().shape()
```