FrOSCon
Automatisierter Flugzeugvorentwurf mit Python

Michael Petsch

Deutsches Zentrum für Luft- und Raumfahrt e.V. (**DLR**)
Institut für Bauweisen und Strukturtechnologie (BT)
Pfaffenwaldring 38-40 | 70569 Stuttgart
michael.petsch@dlr.de
Tel.: +49 711 6862 368

DLR

FrOSCon
Free and Open Source Software Conference

Automatisierter Flugzeugvorentwurf mit Python

1) Flugzeugvorentwurf Allgemein
2) Flugzeugvorentwurf Prozesskette
3) Open Source Toolumgebung zur Strukturauslegung
4) Fazit

1) Flugzeugvorentwurf Allgemein

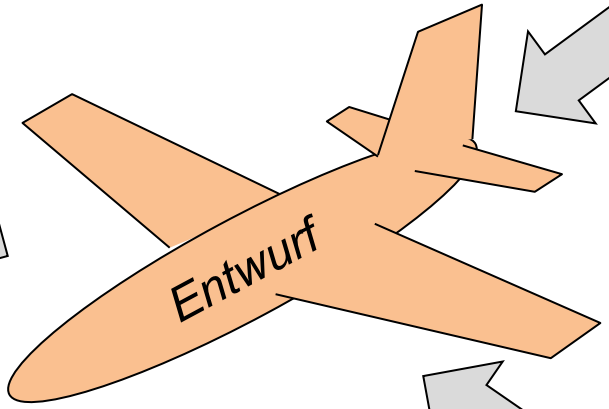
Unterschiedlichste Anforderungen an Luftfahrzeuge

Aerodynamik

[1]

Nutzlast

[2]



Struktur

[3]

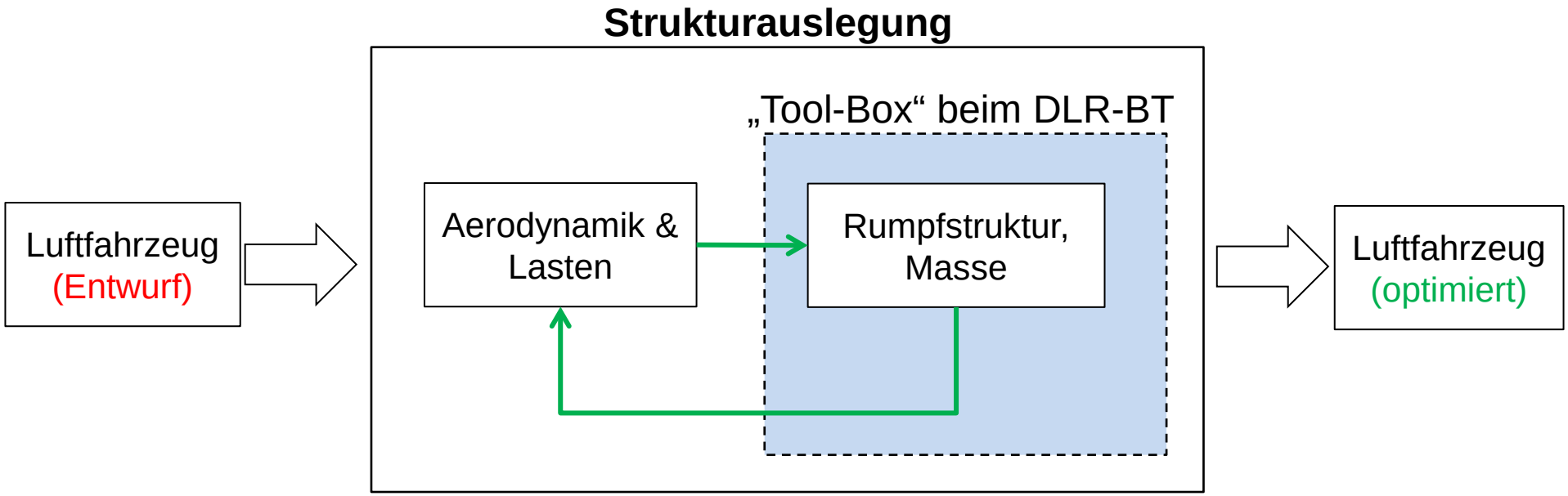
Flugleistung

[4]

[1] https://upload.wikimedia.org/wikipedia/commons/7/79/DG1000_glider_crop.jpg
[2] http://assets.geo.de/div/image/61309/02_columbus_popup.jpg
[3] http://www.ultralightnews.com/_frame/elazair-electric-powered-ultralight-aircraft.jpg
[4] http://www.extraaircraft.com/images/defaults/aircraft/330SC/slideshow/1.jpg

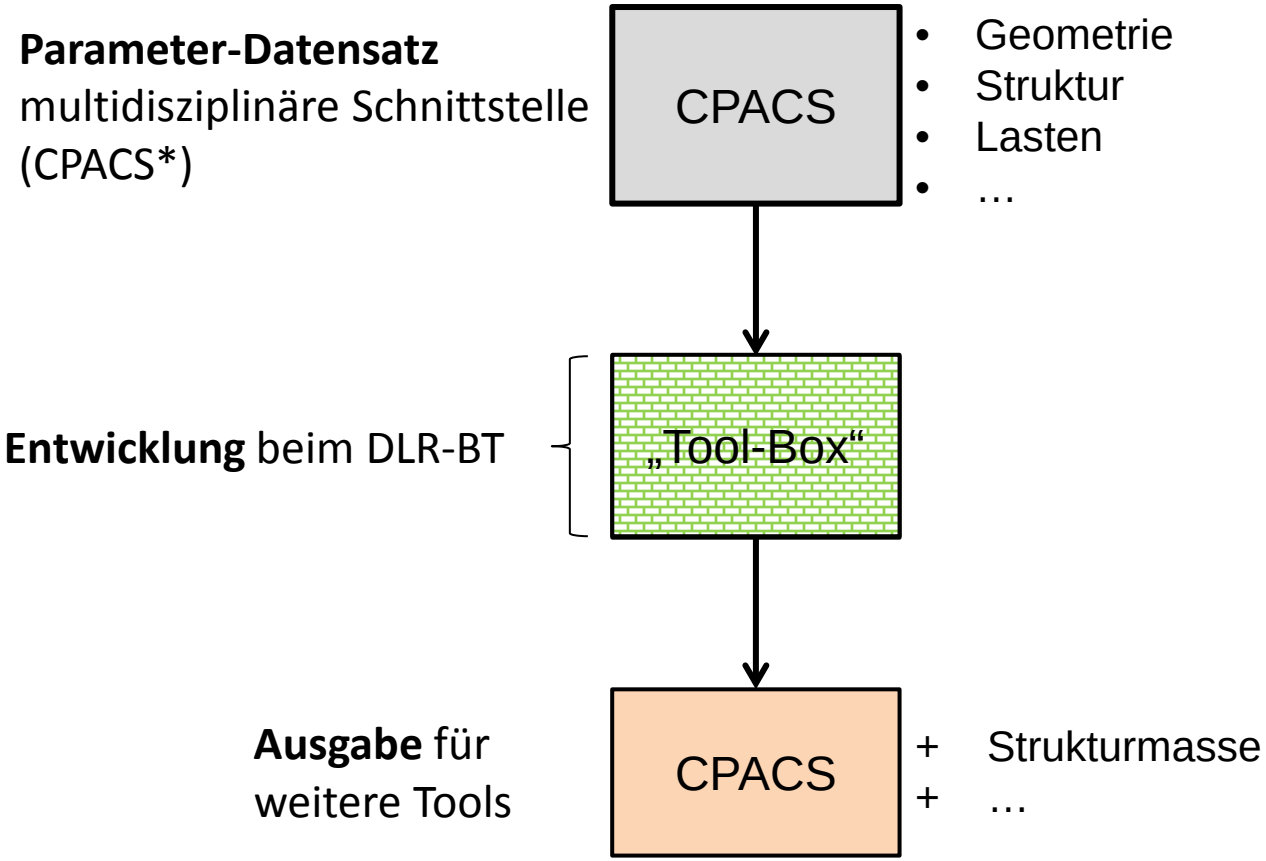
2) Flugzeugvorentwurf Prozesskette

Iterativer Prozess



2) Flugzeugvorentwurf Prozesskette

Prozesskette am DLR



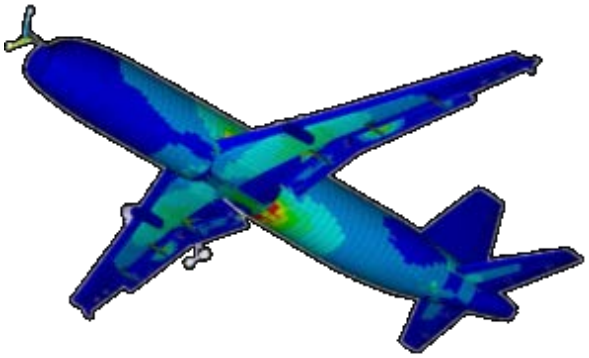
* CPACS = Common Parametric Aircraft Configuration Schema

2) Flugzeugvorentwurf Prozesskette

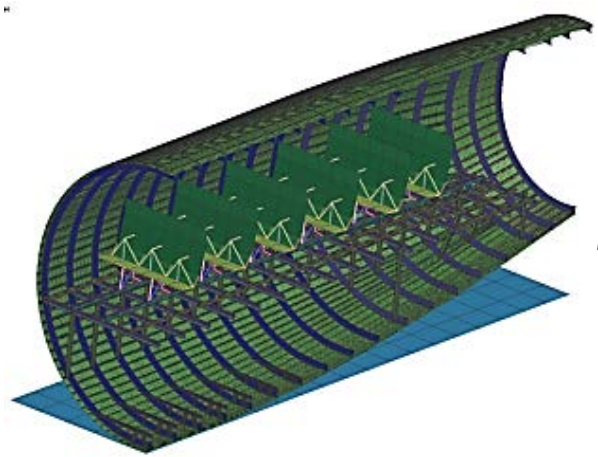
Verschiedene Simulations-Anforderungen

Vorhandene Prozessketten:

- TRAFUMO, AC-CRASH,…
- Kaum gemeinsame Schnittstellen
- Kommerzielle, langsame Skriptsprachen (z.B. APDL der FE-Software ANSYS)
- Unübersichtlich, unflexibel

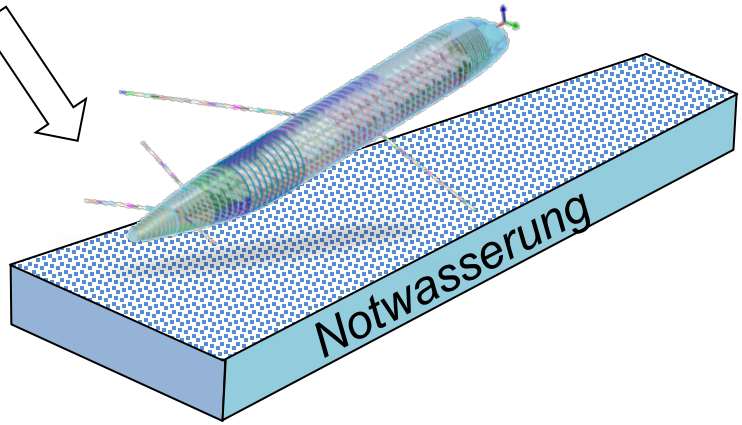


Statische Strukturauslegung



Crash

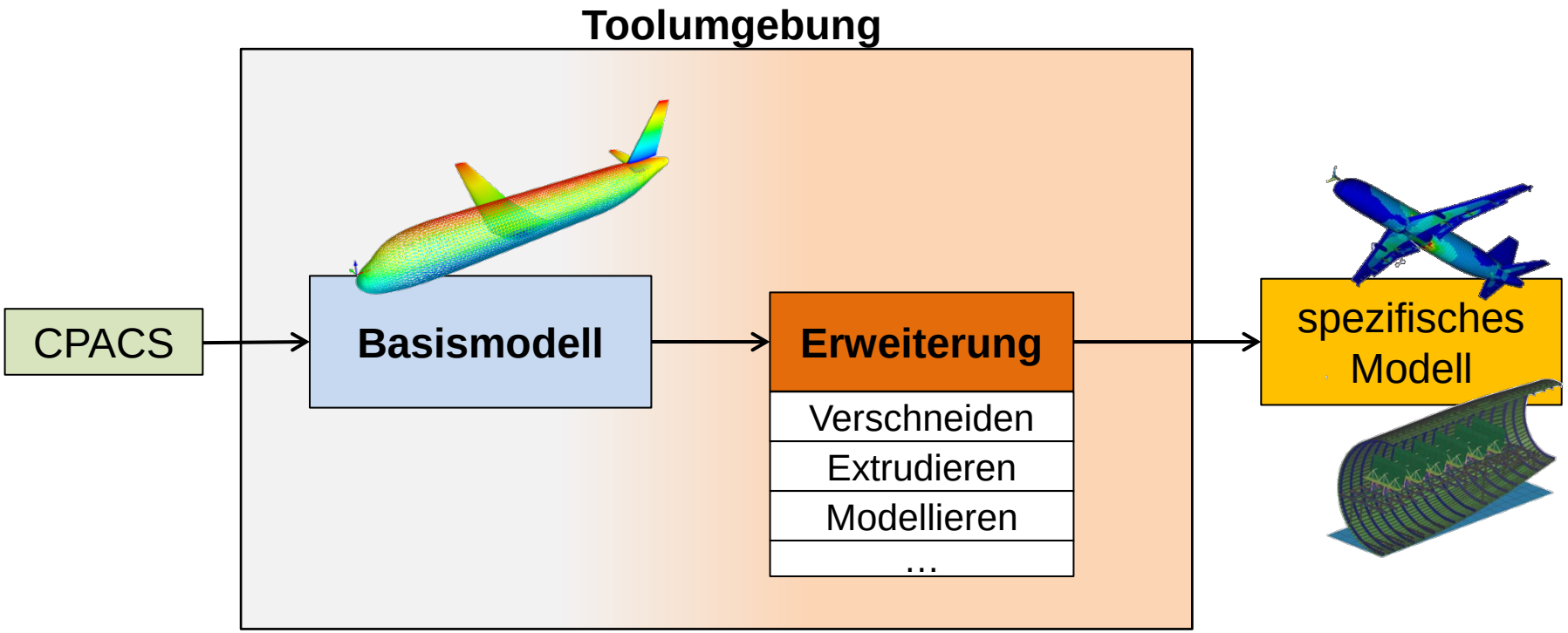
CPACS



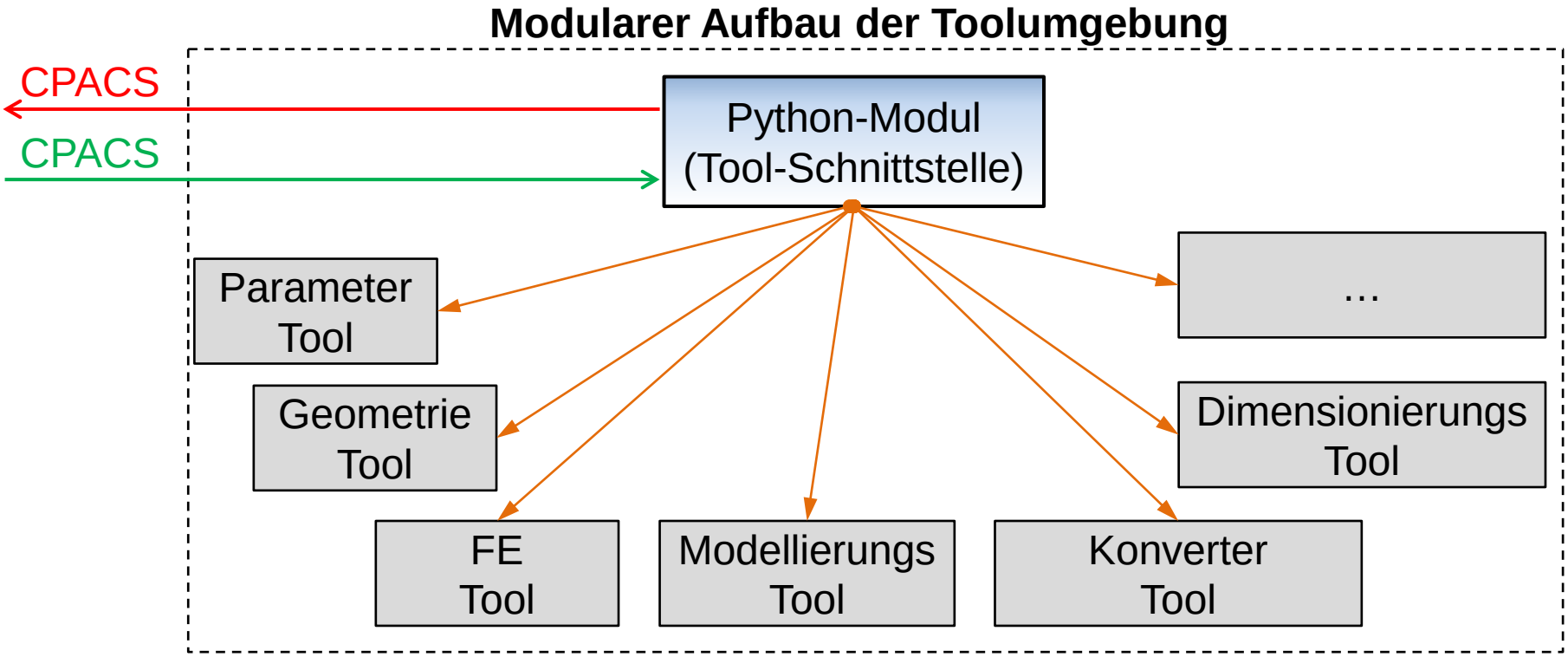
2) Flugzeugvorentwurf Prozesskette

Neue Umsetzung in Python:

- Ein Basismodell
- Eine Toolumgebung (Open Source)
- Modulare Anpassung



3) Open Source Toolumgebung zur Strukturauslegung



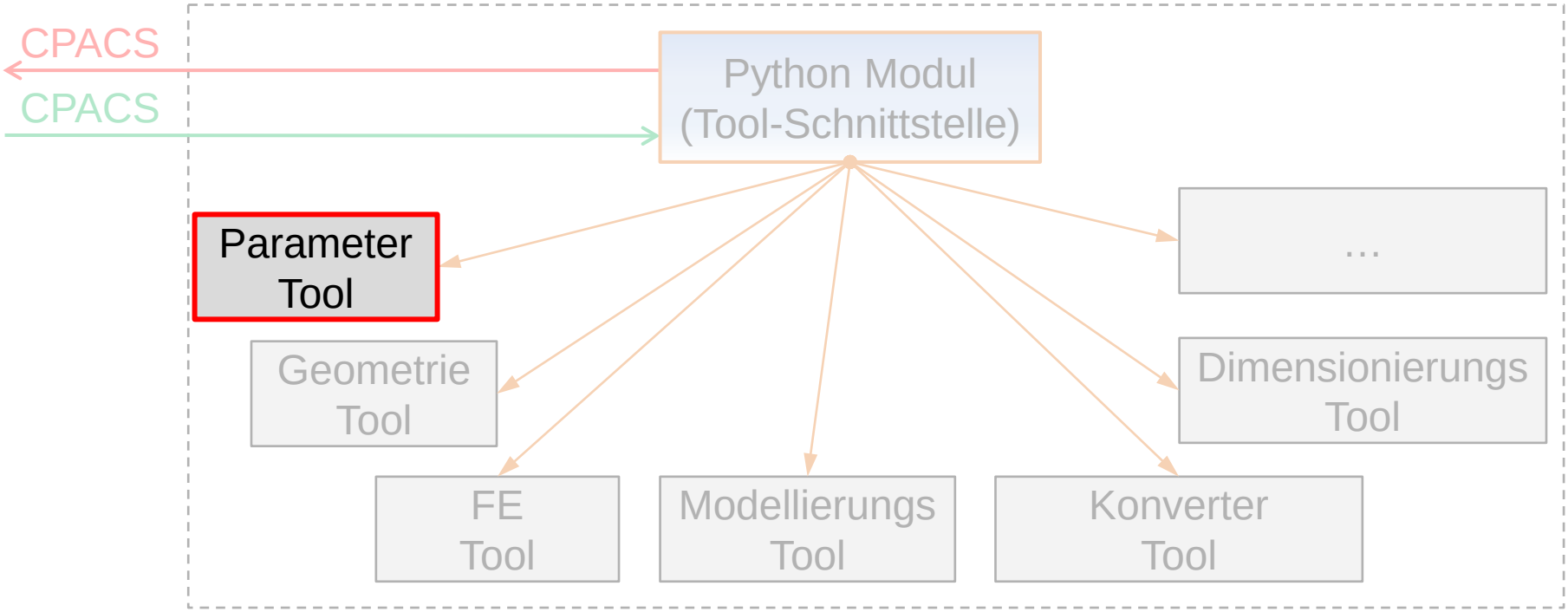
Vorteile:

- Modular (Übersicht, Wartung)
- Einheitliche Python Schnittstelle
- Eigene/lizenzfreie Tools

Nachteile:

- Erhöhter Wartungs- und Programmieraufwand
- Zusätzliche Dokumentationen

3) Open Source Toolumgebung zur Strukturauslegung



3) Open Source Toolumgebung zur Strukturauslegung

Parameter –Tool

Ein-/Auslesen und Validieren der CPACS-Daten (XML-Datei)

3) Open Source Toolumgebung zur Strukturauslegung

Parameter –Tool

Objektorientierte XML-Daten-Verarbeitung

Ast
Objekt(lxml)

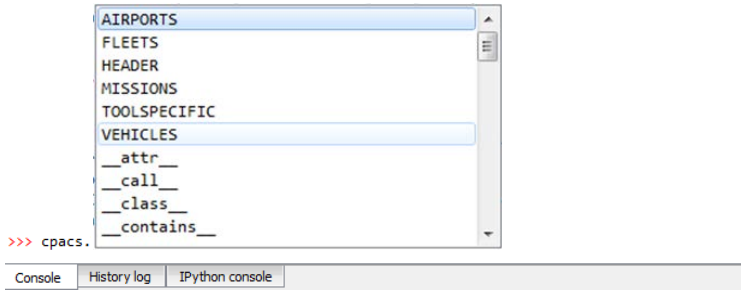
Methoden
- Unteräste iterieren
- Unteräste Durchsuchen
- Unteräste Visualisieren
- Unteräste in Tabelle konvertieren (pandas.DataFrame)
- Ast/Daten modifizieren
- …

Eigenschaften
- ID/Name/Typ
- Dateninhalt
- Verknüpfte Verzweigungen

3) Open Source Toolumgebung zur Strukturauslegung

Parameter –Tool

Pfad Autovervollständigung



Automatische Konvertierung von XML-Text

```
>>> cpacs.VEHICLES.AIRCRAFT.D150MODELID.FUSELAGES.D150_VAMP_FL1.STRUCTURE.FRAMES.C01.FRAMEPOSITION.POSITIONX()
0.96
```

Console | History log | IPython console

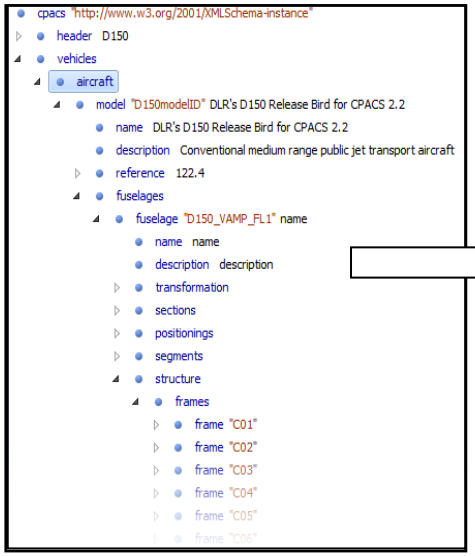
```
>>> cpacs.VEHICLES.PROFILES.WINGAIRFOILS.WINGBODYTAIL_HORIZONTALTAILPLANE_SEC1_EL1_PRO.POINTLIST.X()
[0.999985606, 0.989805118, 0.979626471, 0.969449664, 0.95927467, 0.949101445, 0.938929927, 0.928760022, 0.91859164 ...
```

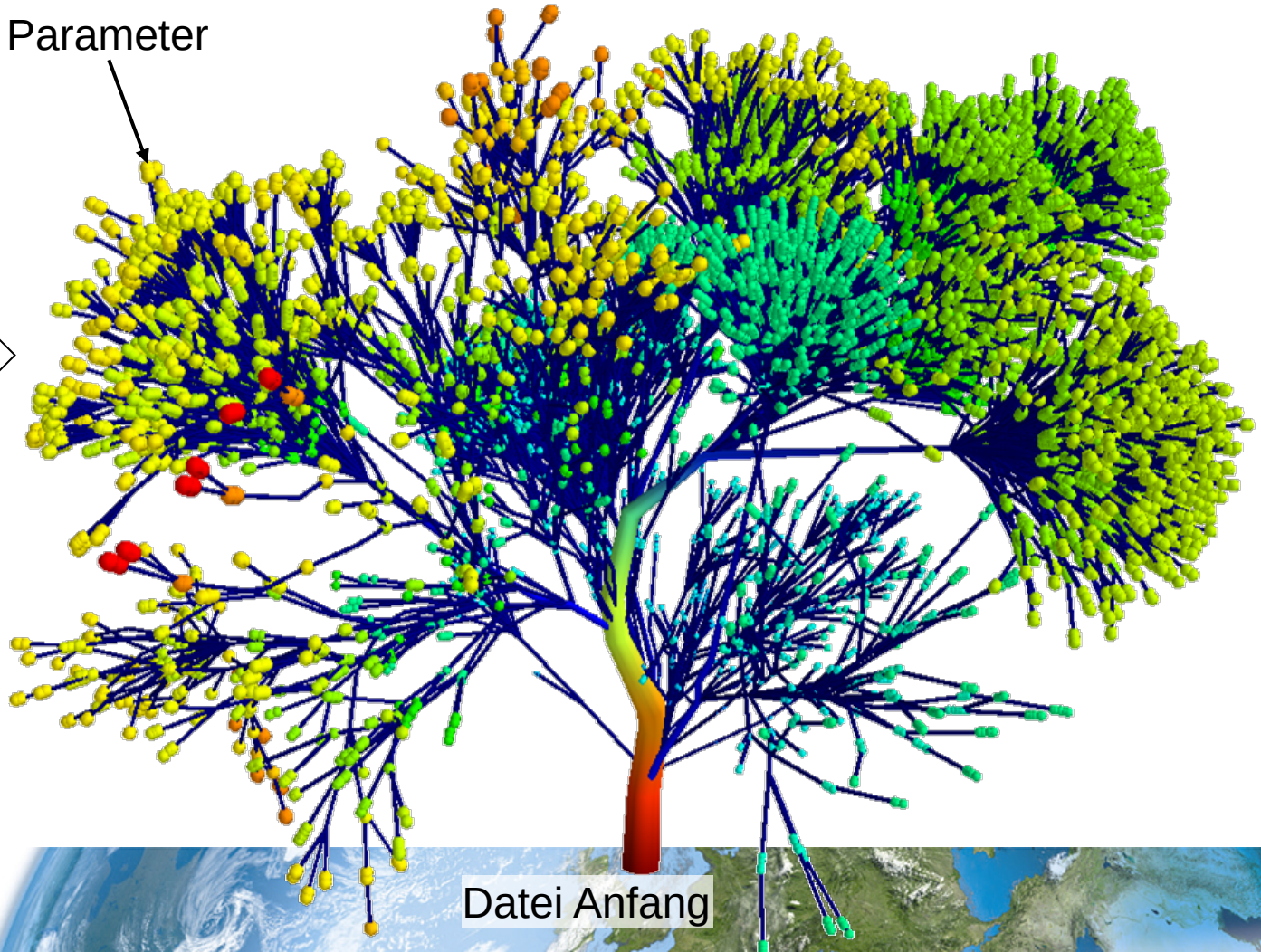
Console | History log | IPython console

3) Open Source Toolumgebung zur Strukturauslegung

Parameter –Tool

XML-Daten Hierarchie visualisieren





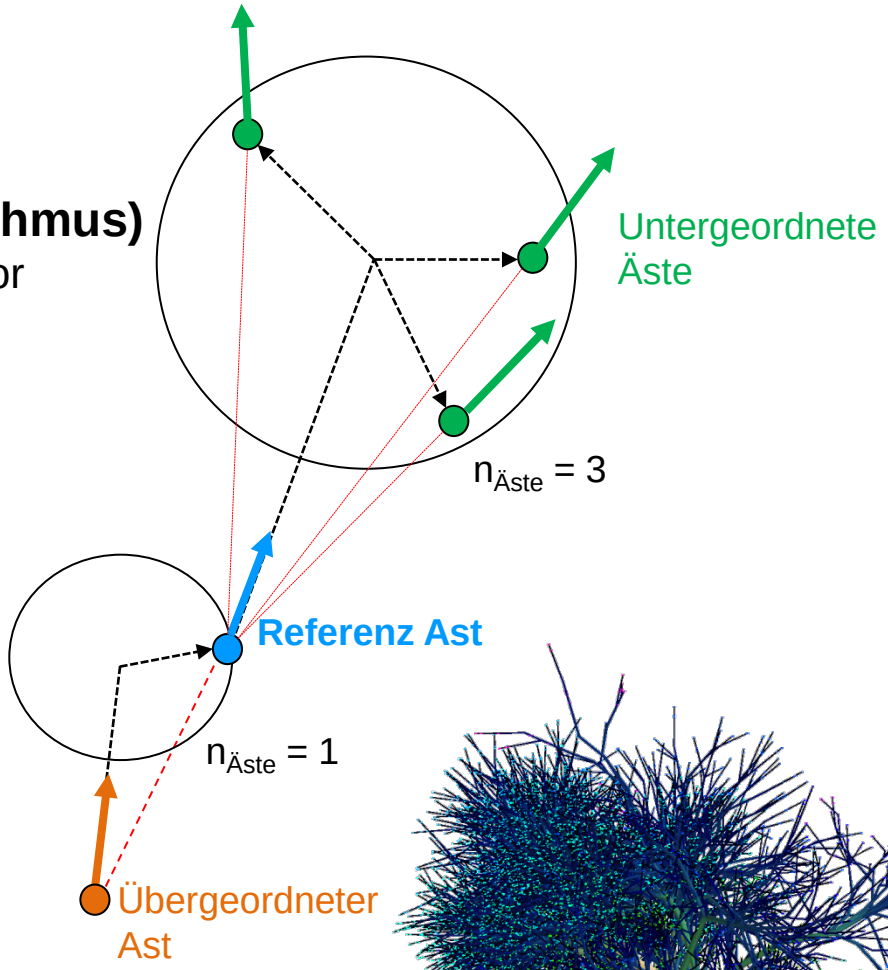
FrOSCon Free and Open Source Software Conference

3) Open Source Toolumgebung zur Strukturauslegung

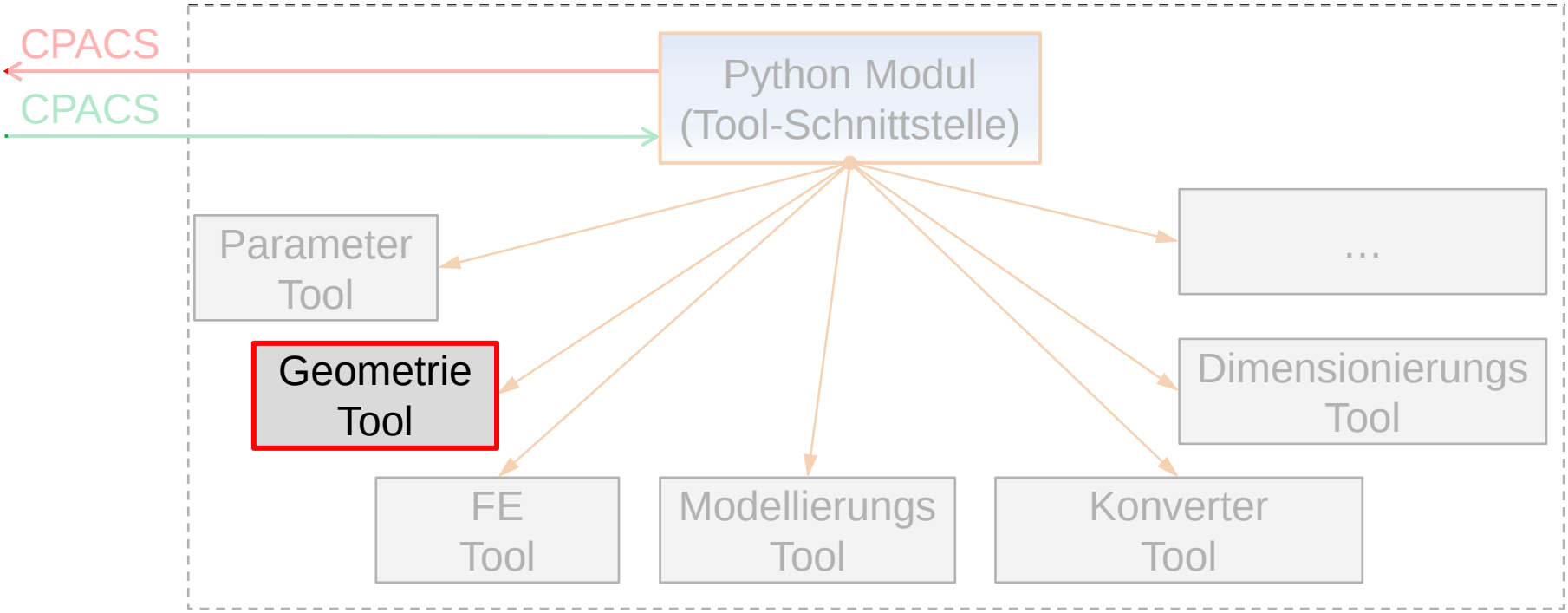
Parameter –Tool

XML-Visualisierung (Baum-Algorithmus)

- Rekursive Zuweisung von Punkt/Vektor
- Relativ zu übergeordneten Objekt
- Skalierung je nach Anzahl Unteräste
- Parameter zur Variation



3) Open Source Toolumgebung zur Strukturauslegung

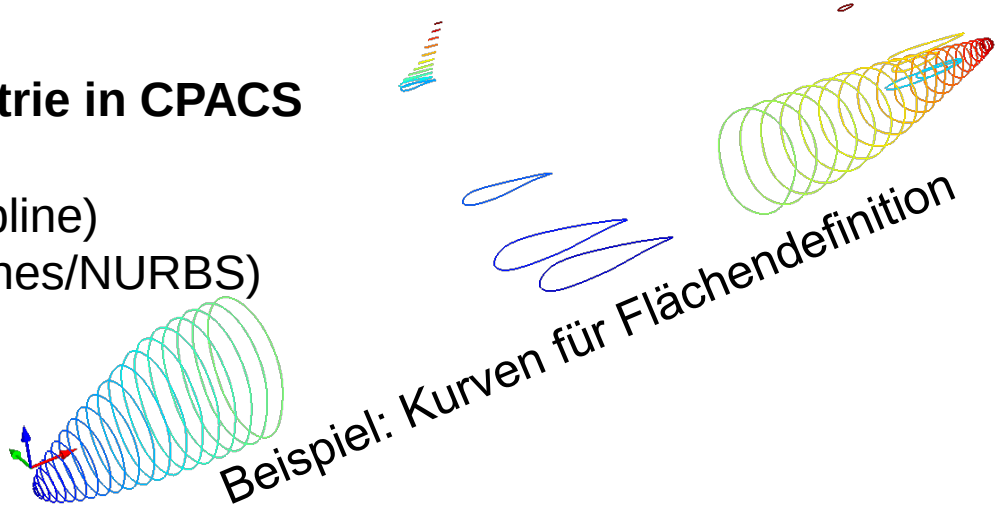


3) Open Source Toolumgebung zur Strukturauslegung

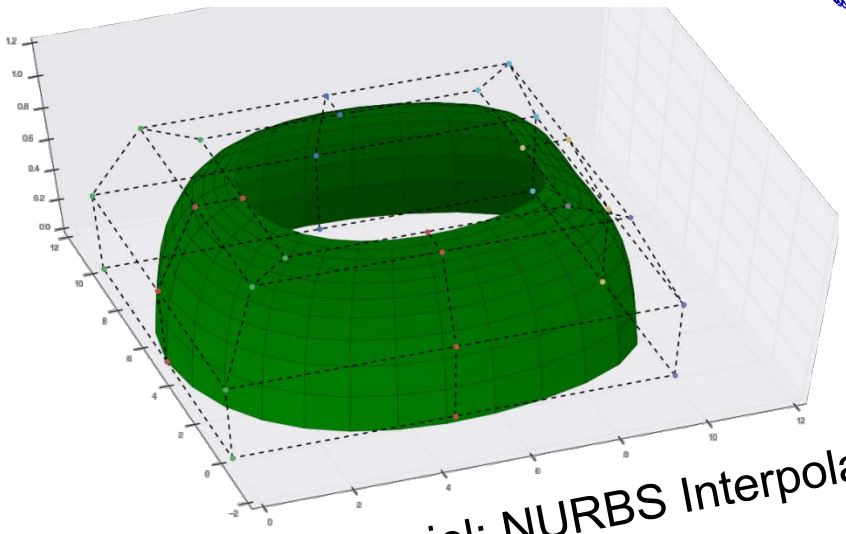
Geometrie –Tool

Beschreibung Oberflächen-Geometrie in CPACS

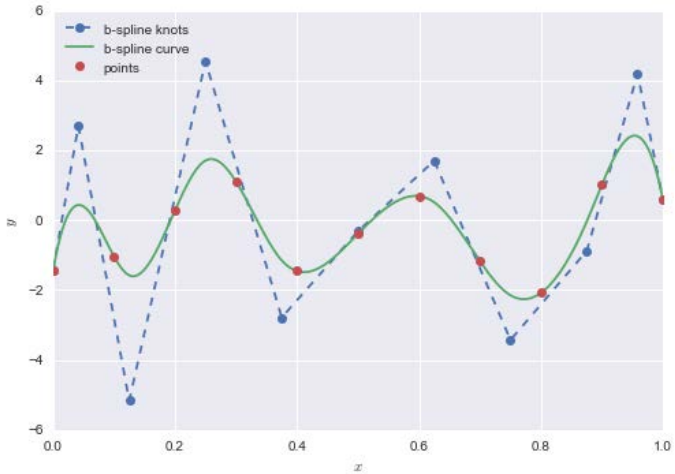
1. Aufpunkte für Kurven
2. Kurveninterpolation (Kubisch/B-Spline)
3. Flächeninterpolation (Coons-Patches/NURBS)



Beispiel: Kurven für Flächendefinition



Beispiel: NURBS Interpolation



Beispiel: B-Spline Interpolation

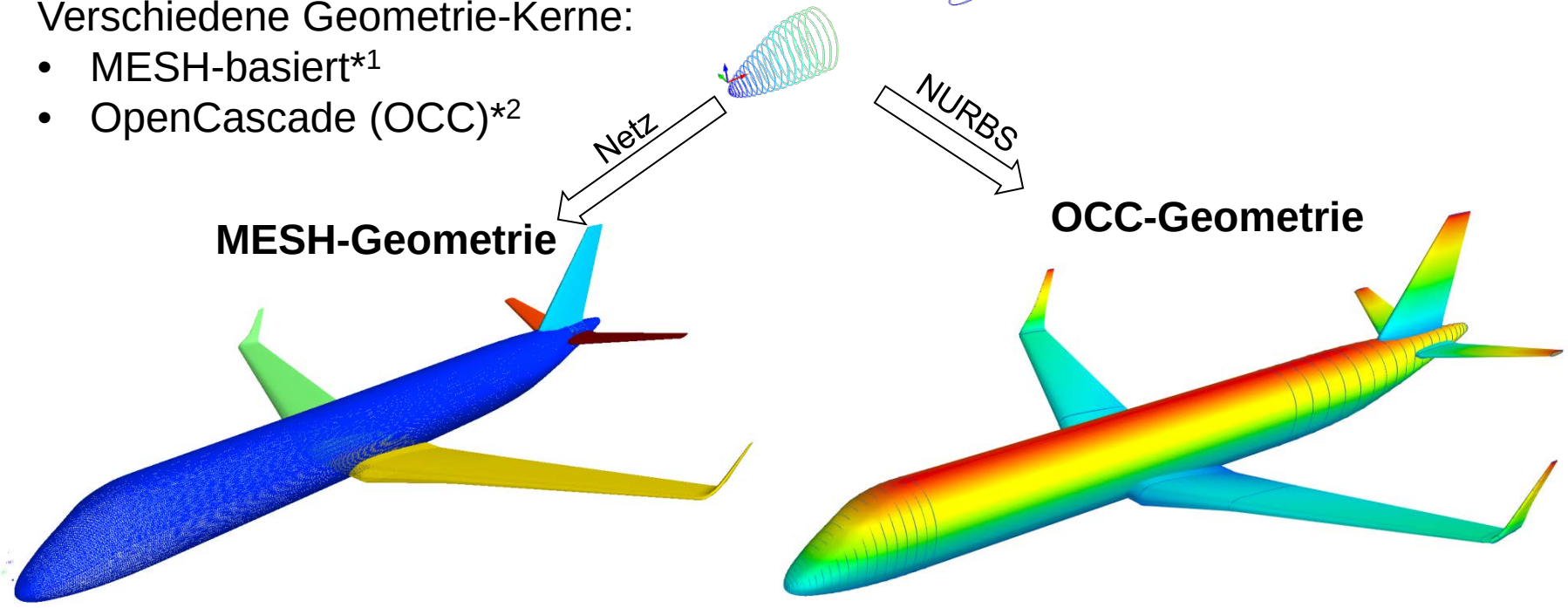
*1 Eigene Geometrie Algorithmen
*2 Über pythonOCC

3) Open Source Toolumgebung zur Strukturauslegung

Geometrie –Tool

Verschiedene Geometrie-Kerne:

- MESH-basiert*[1]
- OpenCascade (OCC)*[2]



- ✓ Schnelle Verschneidungen
- ✓ Variable Netzfeinheit
- – Mäßige Gradienten Qualität

- ✓ Gute Geometriequalität
- ✓ Viele Geometrieoperationen
- – Langsam bei aufwändigen Verschnitten

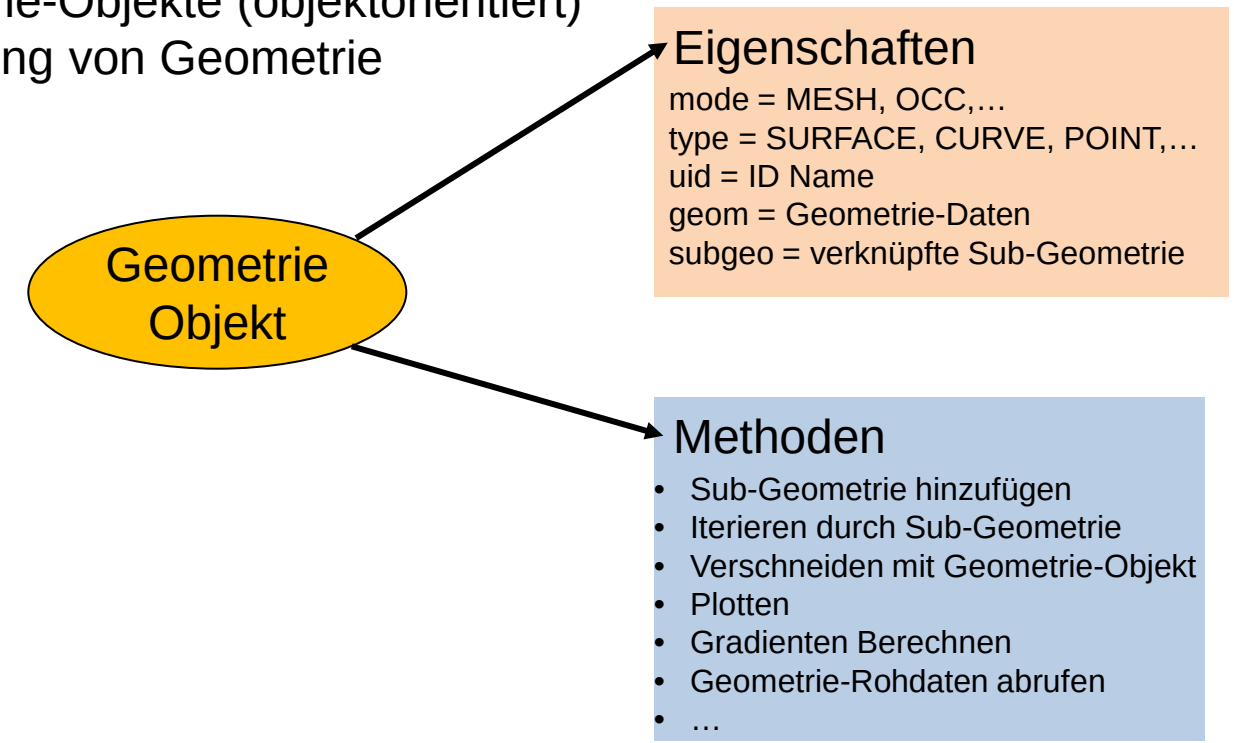
*[1] Eigene Geometrie Algorithmen
*[2] Über pythonOCC

3) Open Source Toolumgebung zur Strukturauslegung

Geometrie –Tool

CPACS-Geometrie-Modul:

- Nutzung beider Geometriekerne
- Geometrie-Objekte (objektorientiert)
- Verwaltung von Geometrie



3) Open Source Toolumgebung zur Strukturauslegung

Geometrie –Tool

Objektorientierte Geometrie

- Geometrietyp automatisch erkannt

Geometrie Objekt D

Geometrie Objekt C

Verschneiden

Geometrie Objekt E1

Geometrie Objekt E2

```
In [22]: new_object._intersect(geom.PLANES.STRINGER.CT_04)
Out[22]: [<__main__.Geom_Object at 0x12b23048>, <__main__.Geom_Object at 0xe9d0630>]
```

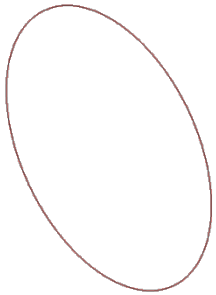
```
In [16]: geom.SURFACES.D150_VAMP_FL1.__type__
Out[16]: 'SURFACE'
```

Geometrie Objekt A

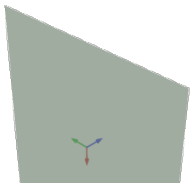
Geometrie Objekt B

Verschneiden

Geometrie Objekt C



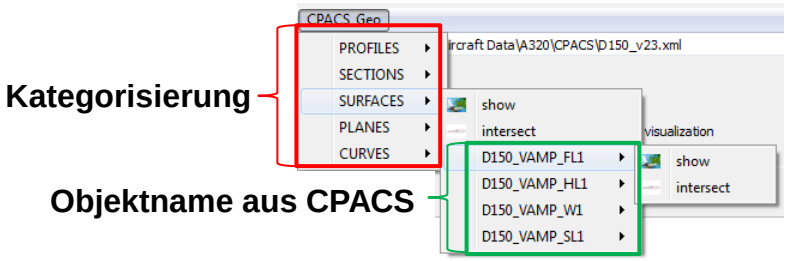
```
In [18]: new_object = geom.SURFACES.D150_VAMP_FL1._intersect(geom.PLANES.FRAME.C04)

In [19]: new_object.__type__
Out[19]: 'CURVE'

In [20]: new_object()
Out[20]: <function __main__.<lambda>>

In [21]: new_object()(0.5)
Out[21]: array([ 1.79      , -0.02143585, -1.86930944])
```



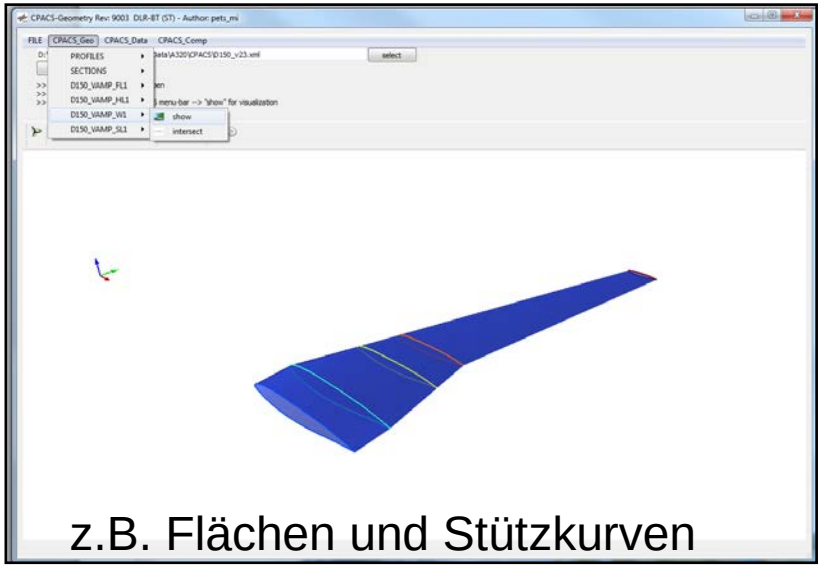
```
In [17]: geom.PLANES.FRAME.C04.__type__
Out[17]: 'PLANE'
```

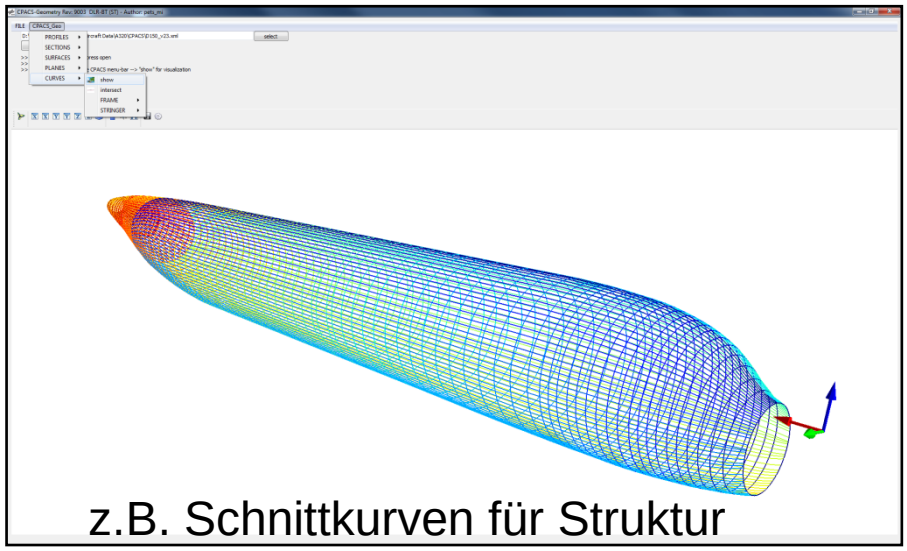

3) Open Source Toolumgebung zur Strukturauslegung

Geometrie –Tool



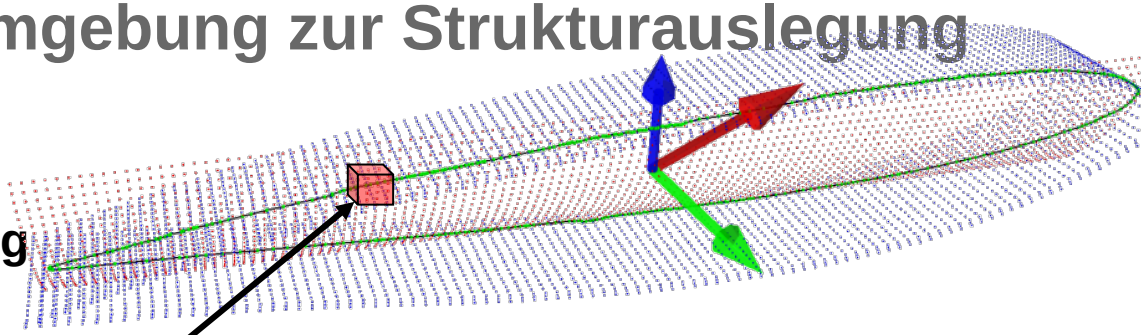
Verwaltung von Geometrie Objekten aus CPACS





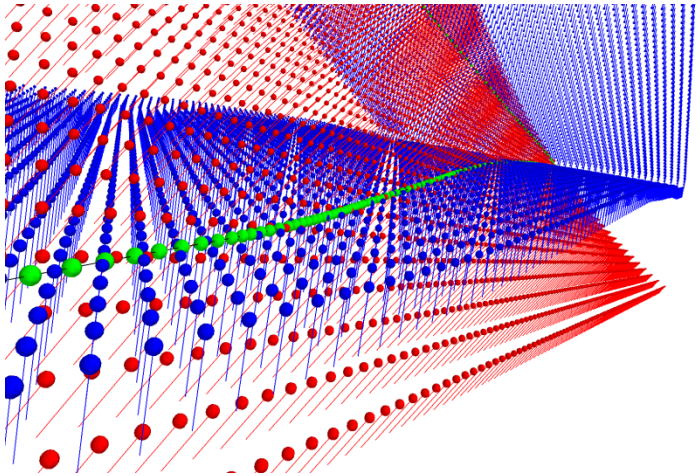
3) Open Source Toolumgebung zur Strukturauslegung

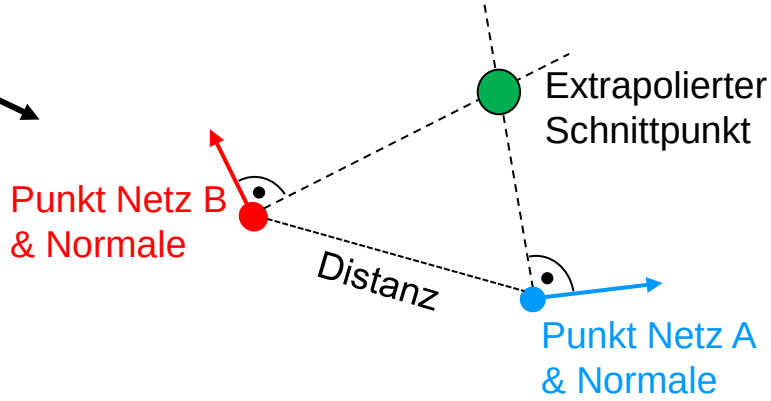
Geometrie –Tool



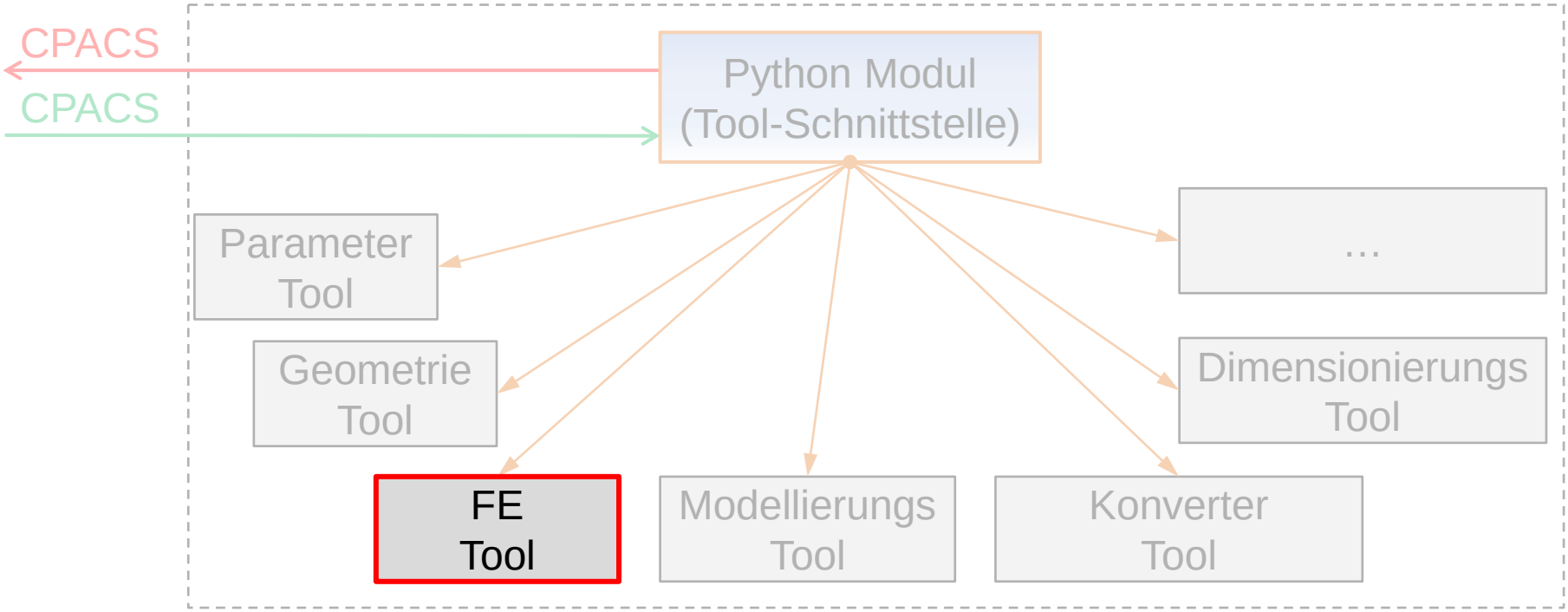
Mesh-basierte Verschneidung

1. Box um Bereich beider Netze
2. Gebiet unterteilen in Boxen
3. Filter Boxen mit Punkte beider Netze
4. Filter Punkte mit Distanz<Toleranz
5. Schnittpunkt Extrapolieren
6. Punkte sortieren/filtern
7. Kurve erstellen





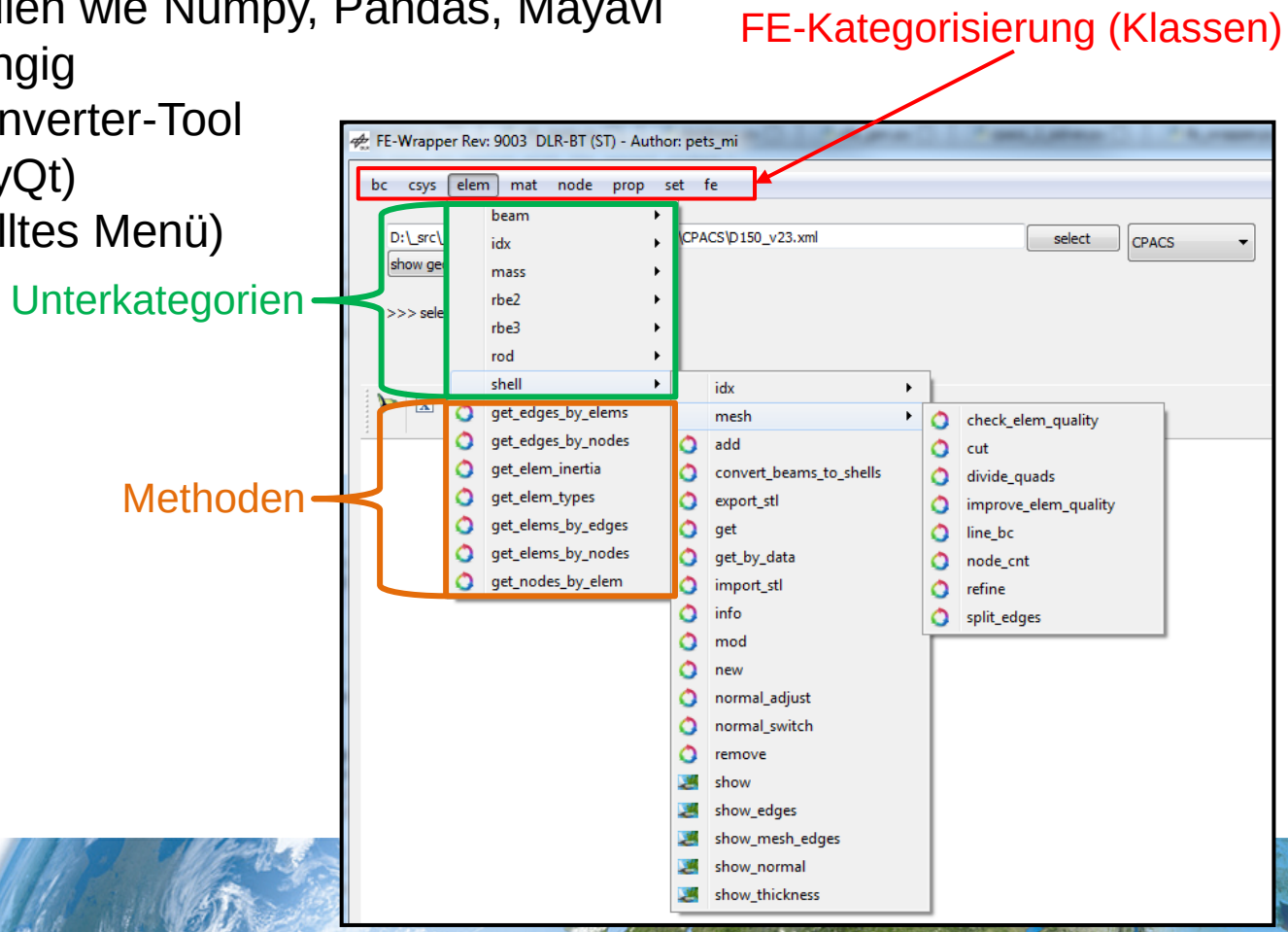
3) Open Source Toolumgebung zur Strukturauslegung



3) Open Source Toolumgebung zur Strukturauslegung

FE – Tool (PreProzessor)

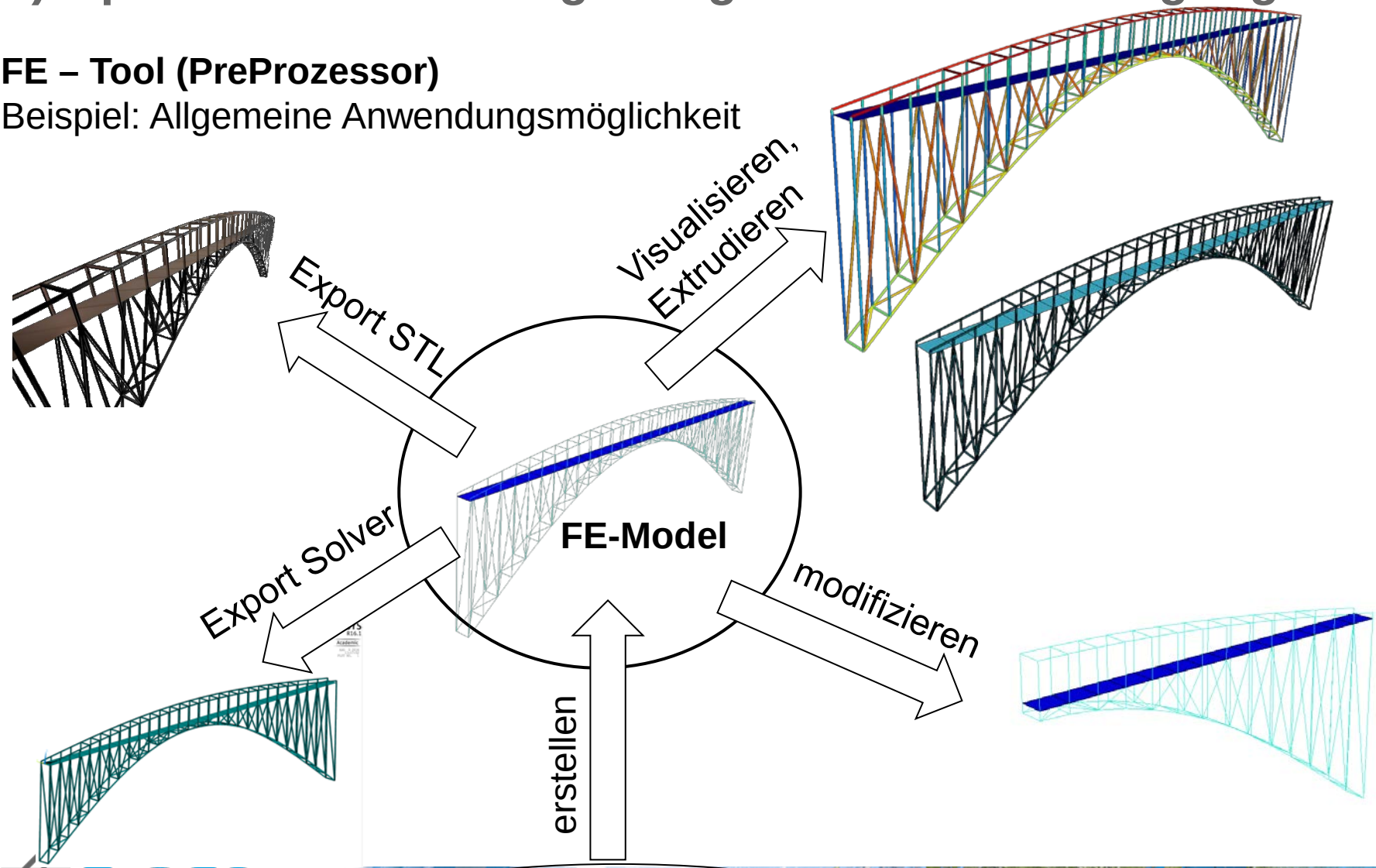
- Finite Elemente (FE) Modellaufbau
- Python basiert (OOP)
- Nutzung von Modulen wie Numpy, Pandas, Mayavi
- FE-Solver unabhängig
- Schnittstelle zu Konverter-Tool
- GUI vorhanden (PyQt)
  (automatisch erstelltes Menü)



3) Open Source Toolumgebung zur Strukturauslegung

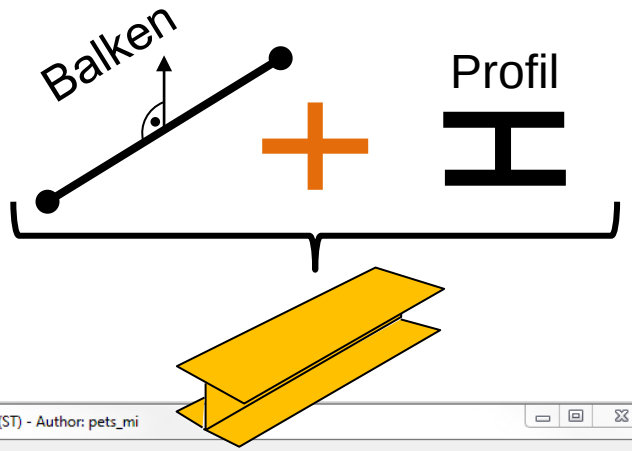
FE – Tool (PreProzessor)

Beispiel: Allgemeine Anwendungsmöglichkeit



3) Open Source Toolumgebung zur Strukturauslegung

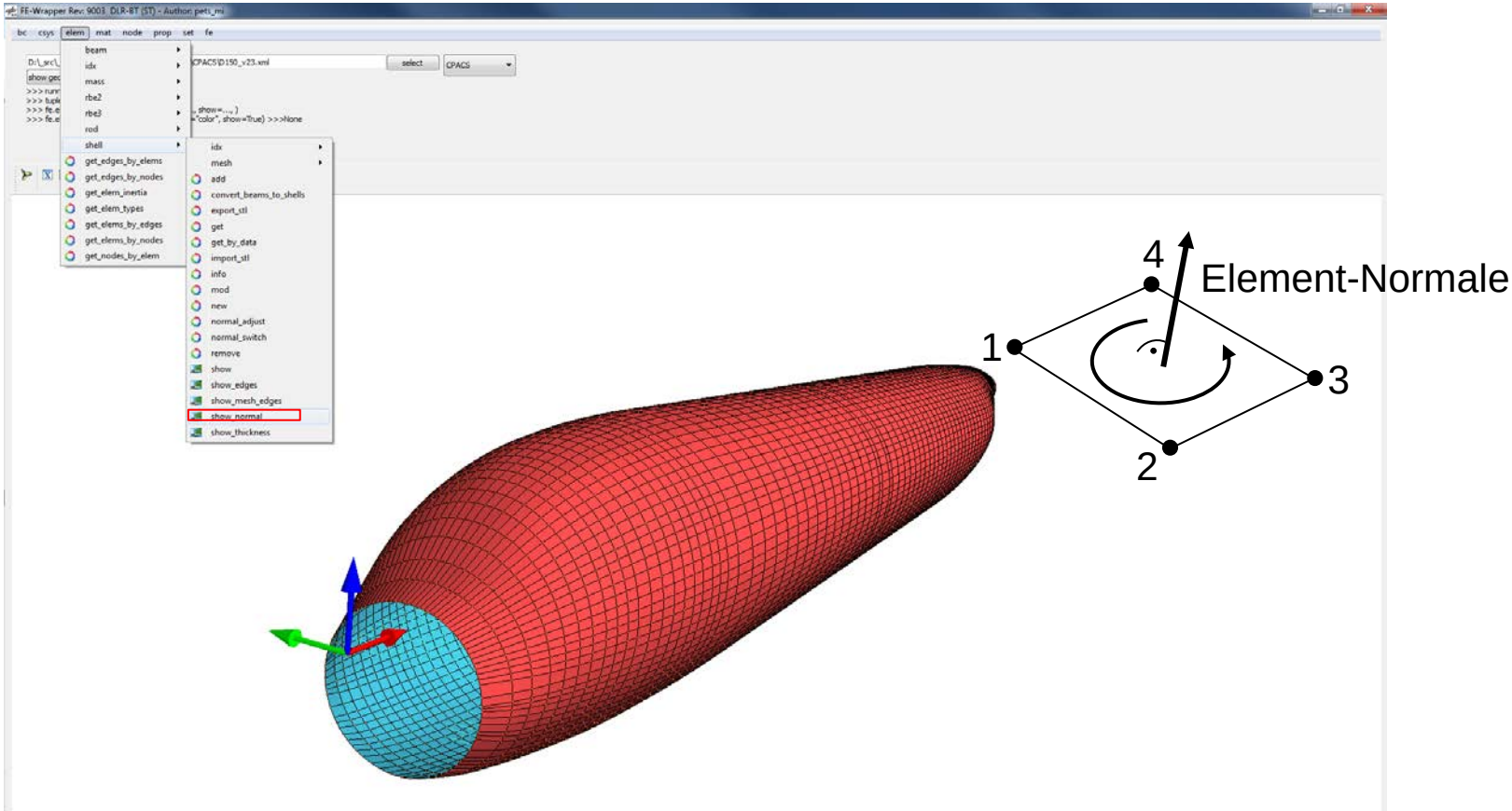
FE – Tool (PreProzessor)
Beispiel: Extrudieren von Balken



3) Open Source Toolumgebung zur Strukturauslegung

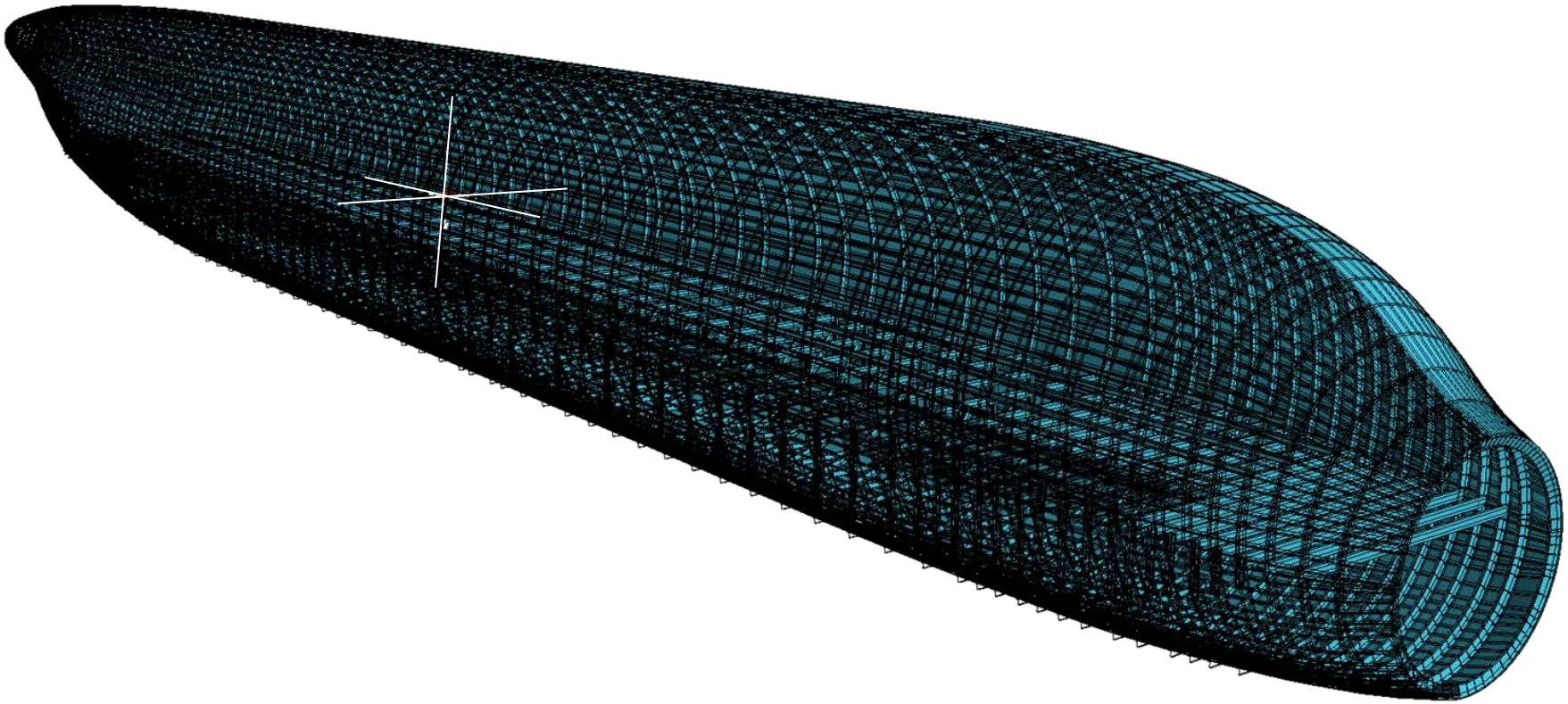
FE – Tool (PreProzessor)

Beispiel: Flächennormalen visualisieren/ausrichten



3) Open Source Toolumgebung zur Strukturauslegung

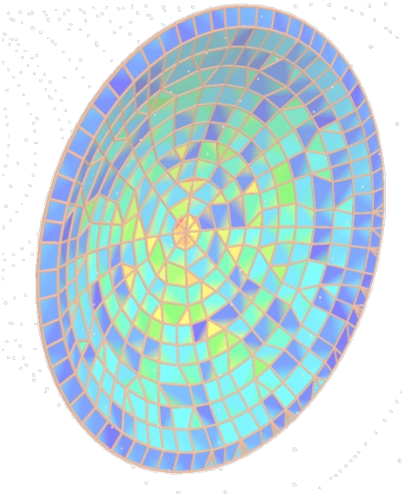
FE – Tool (PreProzessor)
Beispiel: Berechnung von Schwerpunkt, Massenträgheitsmomente,…

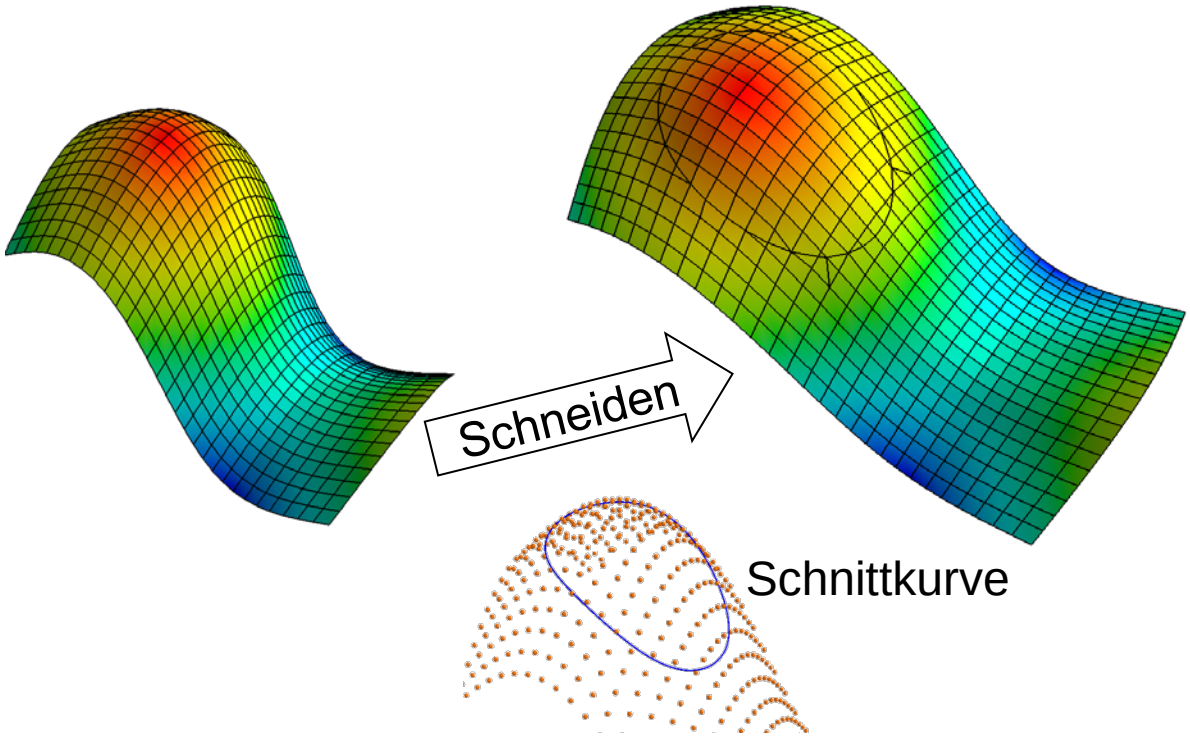


3) Open Source Toolumgebung zur Strukturauslegung

FE – Tool (PreProzessor)
Beispiel: FE-Netz modifizieren

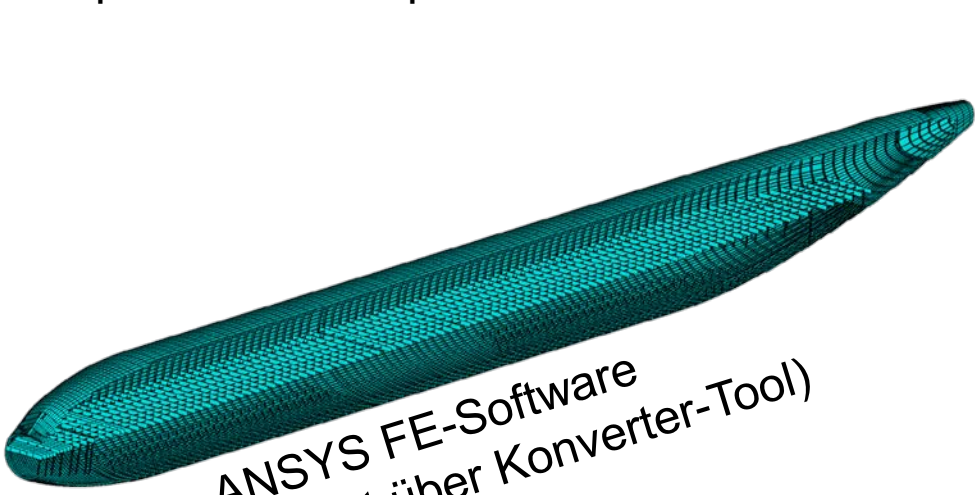
„Vernetzung“
(bisher nur bei vorhandenen Knoten & Knotenverbindungen)



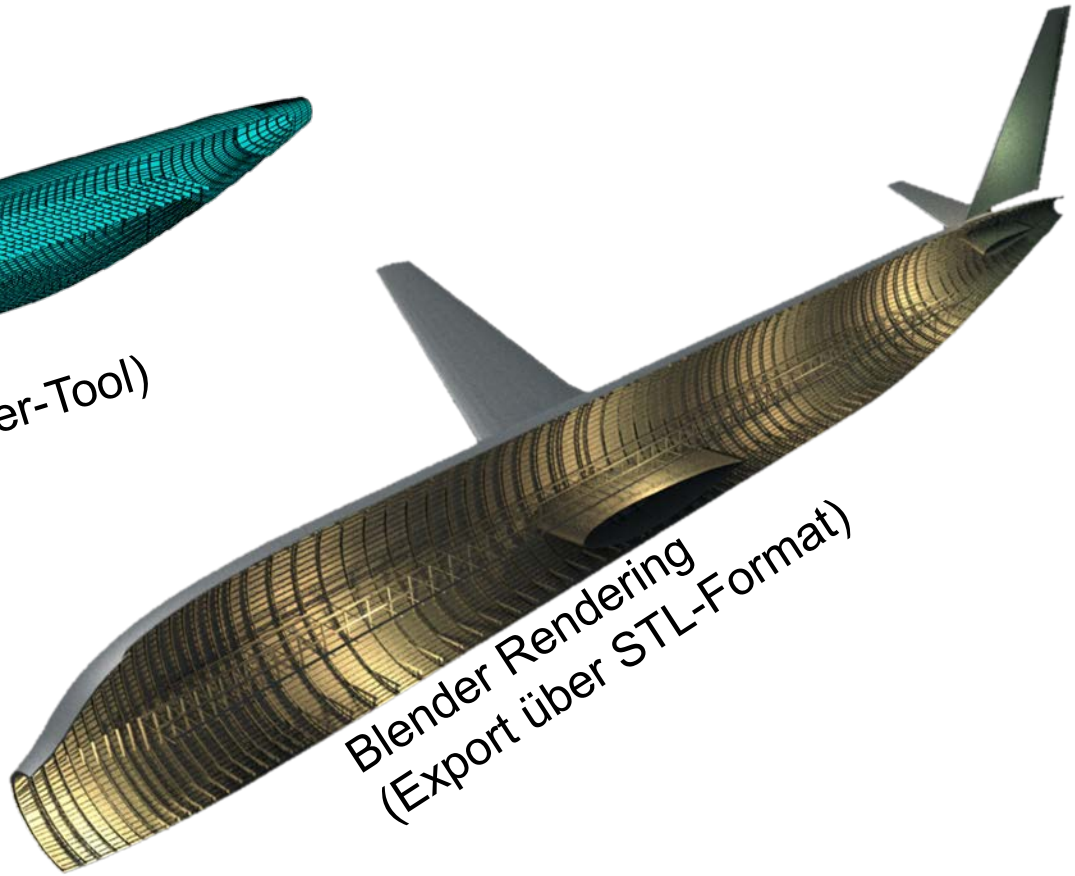


3) Open Source Toolumgebung zur Strukturauslegung

FE – Tool (PreProzessor)
Beispiel: Modell Exportieren

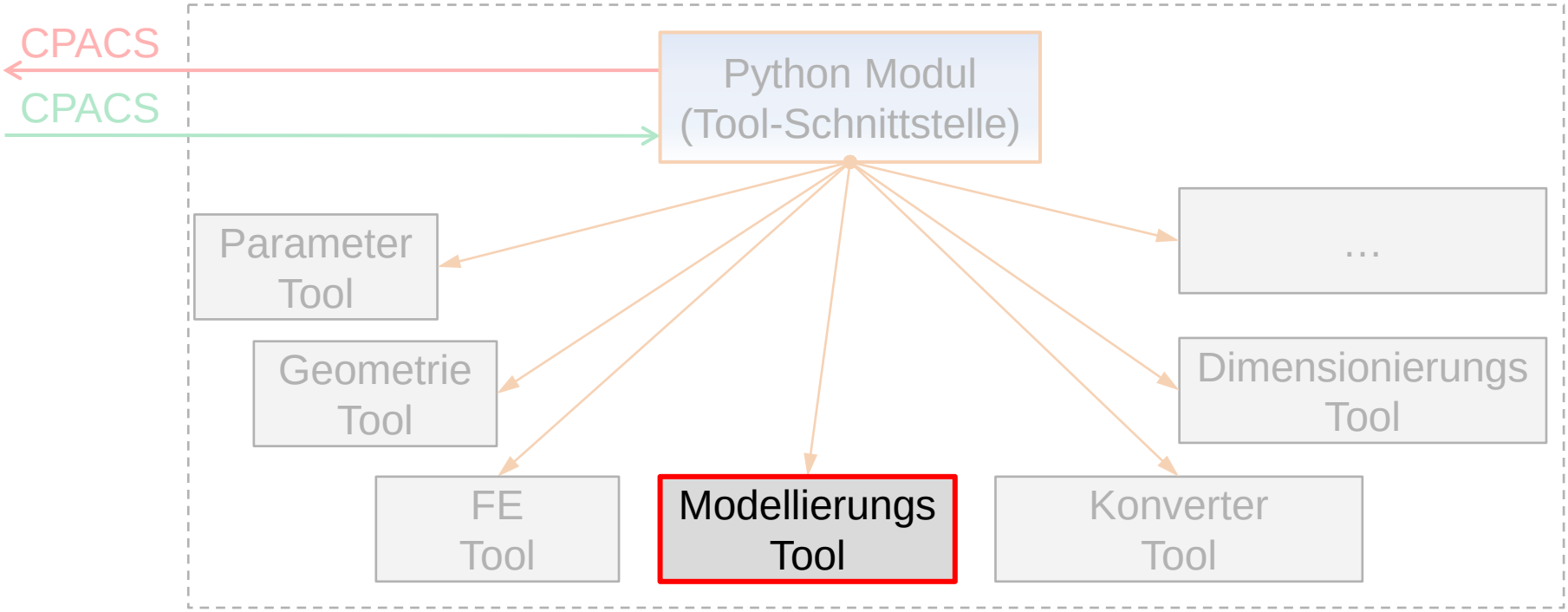


ANSYS FE-Software
(Export über Konverter-Tool)



Blender Rendering
(Export über STL-Format)

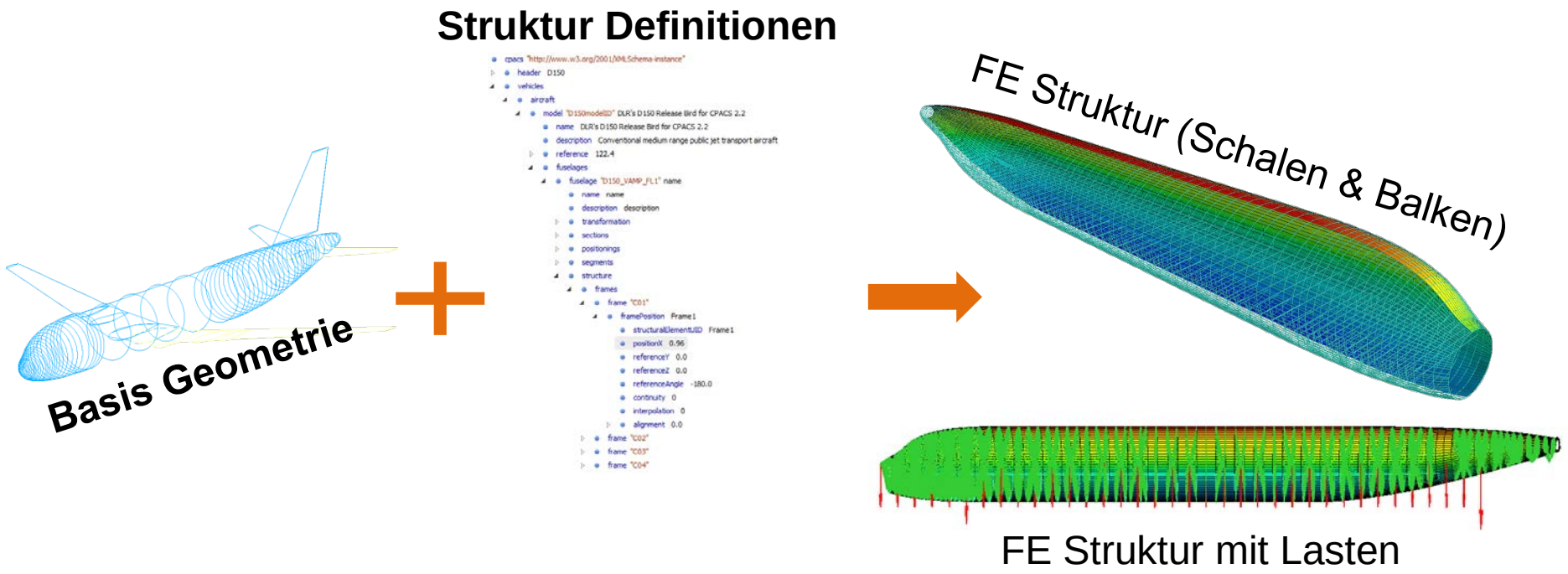
3) Open Source Toolumgebung zur Strukturauslegung



3) Open Source Toolumgebung zur Strukturauslegung

Basis-Modellierungs – Tool

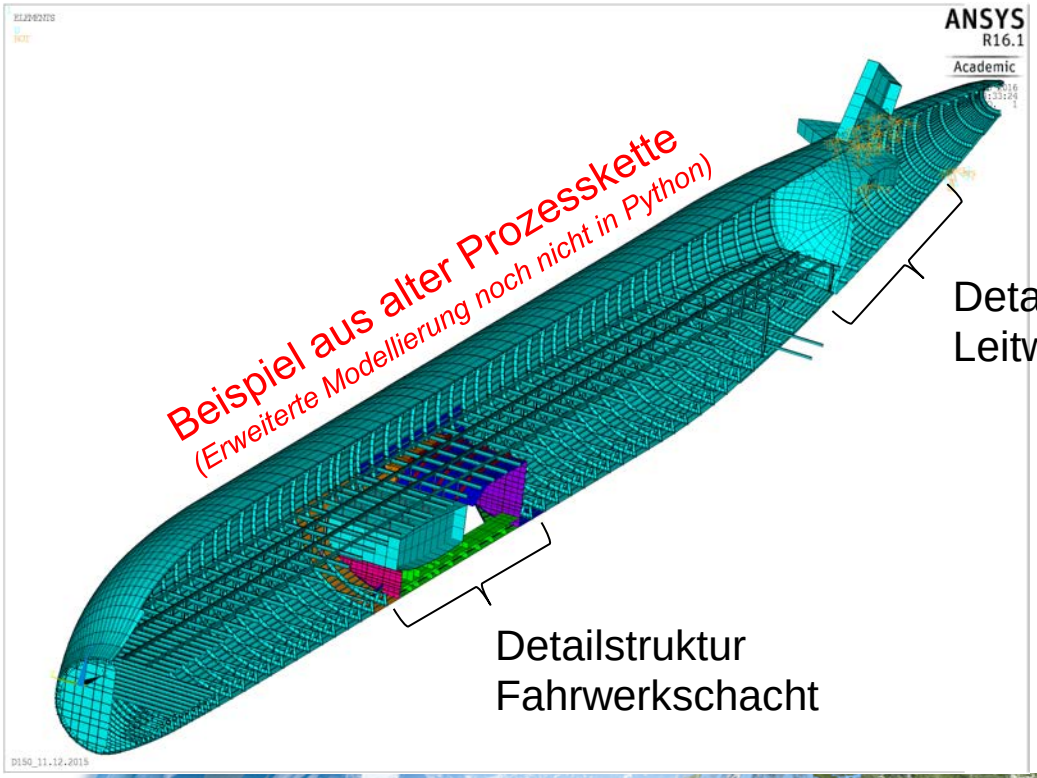
- Nutzung vieler Module (Parameter, Geometrie, FE)
- Algorithmen zur Erstellung der FE-Rumpfstruktur



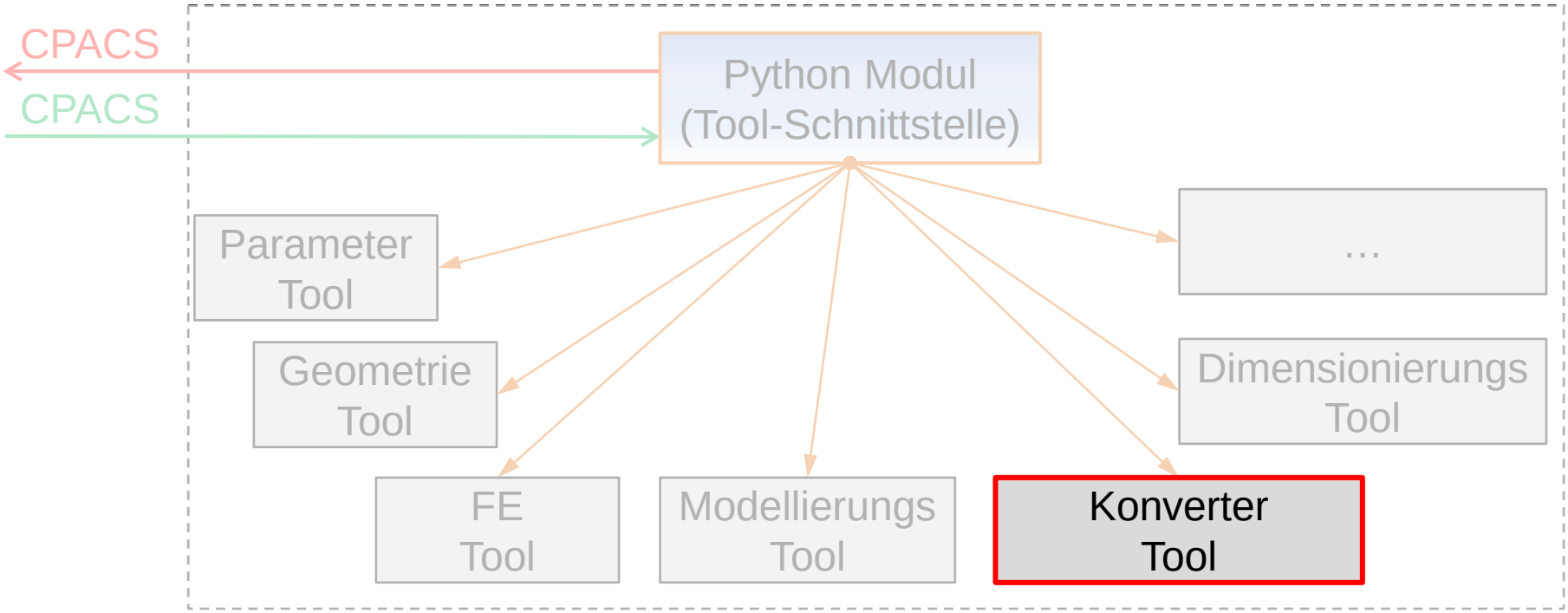
3) Open Source Toolumgebung zur Strukturauslegung

Erweitertes-Modellierungs – Tool

- Detailstruktur modellieren
- Netzverfeinerung
- Ausmodellierung vereinfachter Struktur (z.B. Balken)



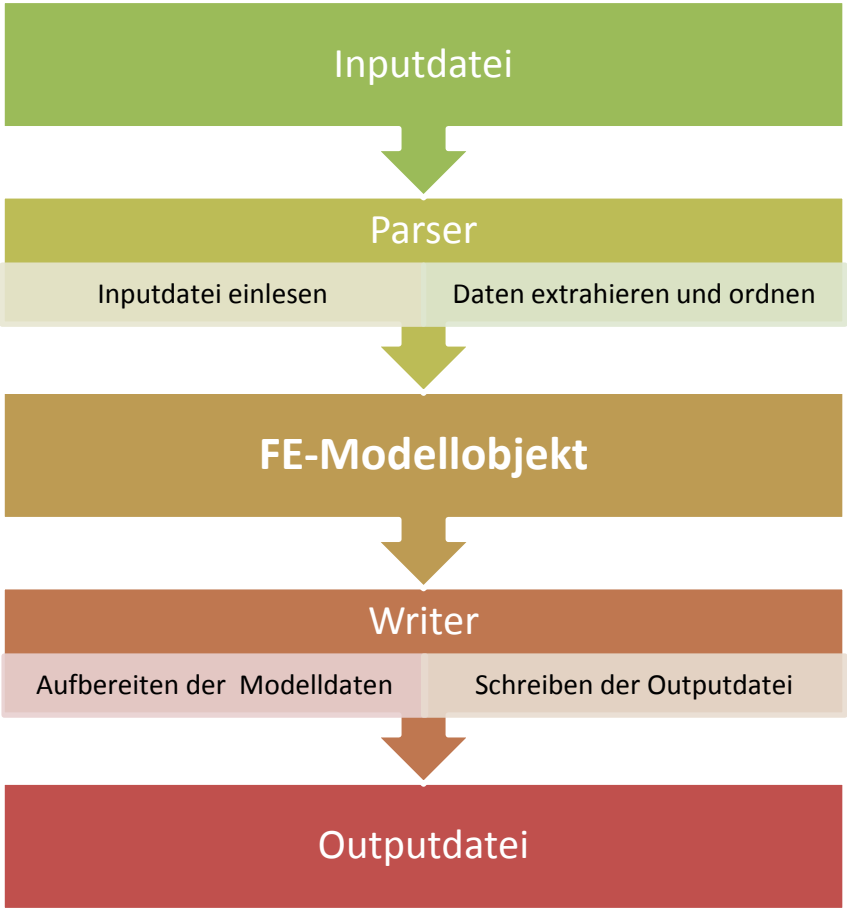
3) Open Source Toolumgebung zur Strukturauslegung



3) Open Source Toolumgebung zur Strukturauslegung

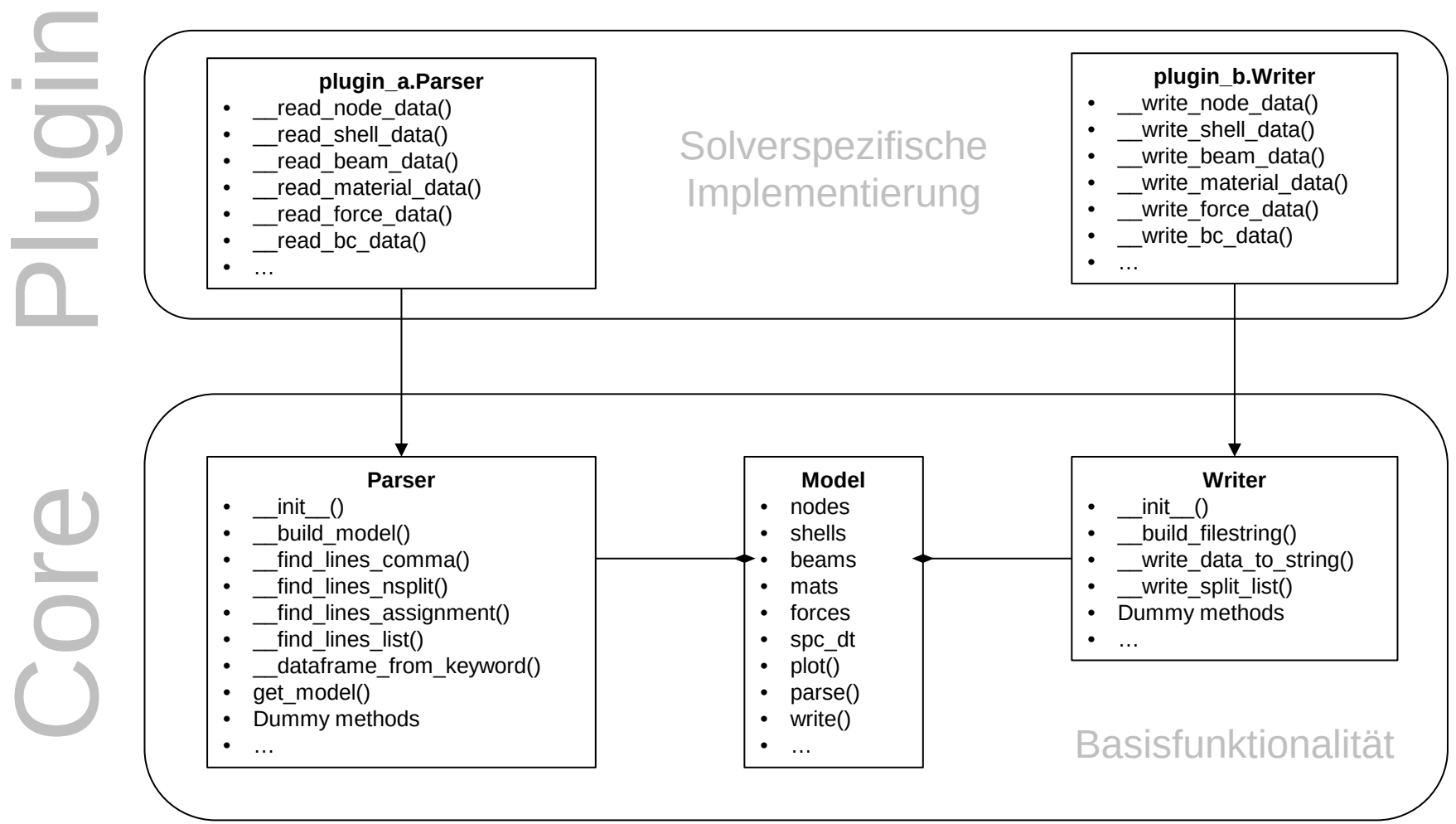
Konverter – Tool

- FE-Daten Konvertieren für verschiedene FE-Codes
  (Ansys, Nastran, Code-Aster (open-source),…)
- Modular & objektorientiert
- Offene interne Datenstruktur
- Tests & HTML-Dokumentation



3) Open Source Toolumgebung zur Strukturauslegung

Konverter – Tool



3) Open Source Toolumgebung zur Strukturauslegung

Konverter – Tool

Modelldatenverarbeitung mit pandas

Beispiel: Daten Einlesen/Aufbereiten mit pandas

```
def __read_force_data(self):

    df = self.__dataframe_from_keyword('forces')



    d = {'N1':'FX', 'N2':'FY', 'N3':'FZ'}
         transl_frame = pd.DataFrame(np.eye(3),
         index=d.values(), columns=d.keys())



    df = df[df.F!=0].join(trans_frame, on='TYPE‘,
         how=‘inner‘).drop('TYPE', axis=1)

    return df.drop(['NEND', 'NINC'], axis=1)
```

	G	TYPE	F	TREF	NEND	NINC
0	5	FX	500	0	NaN	NaN
1	9	FY	-5000	0	NaN	NaN
2	10	FX	500	0	NaN	NaN

	N1	N2	N3
FX	1	0	0
FY	0	1	0
FZ	0	0	1

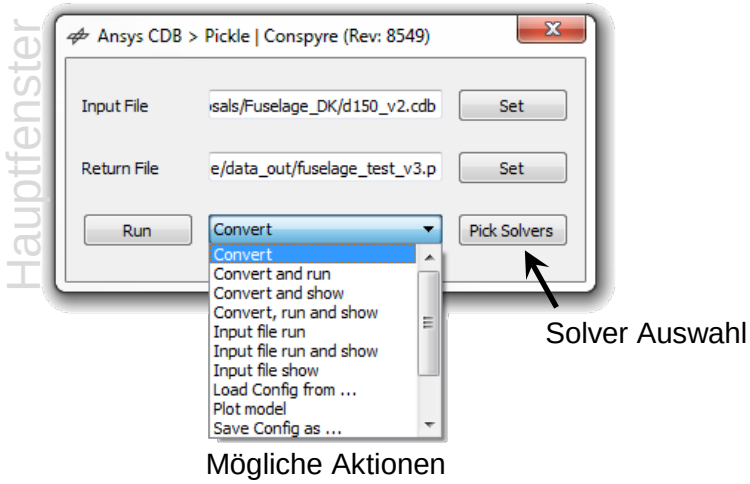
	G	F	TREF	N1	N2	N3
0	5	500	0	1	0	0
1	9	-5000	0	0	1	0
2	10	500	0	1	0	0

3) Open Source Toolumgebung zur Strukturauslegung

Konverter – Tool

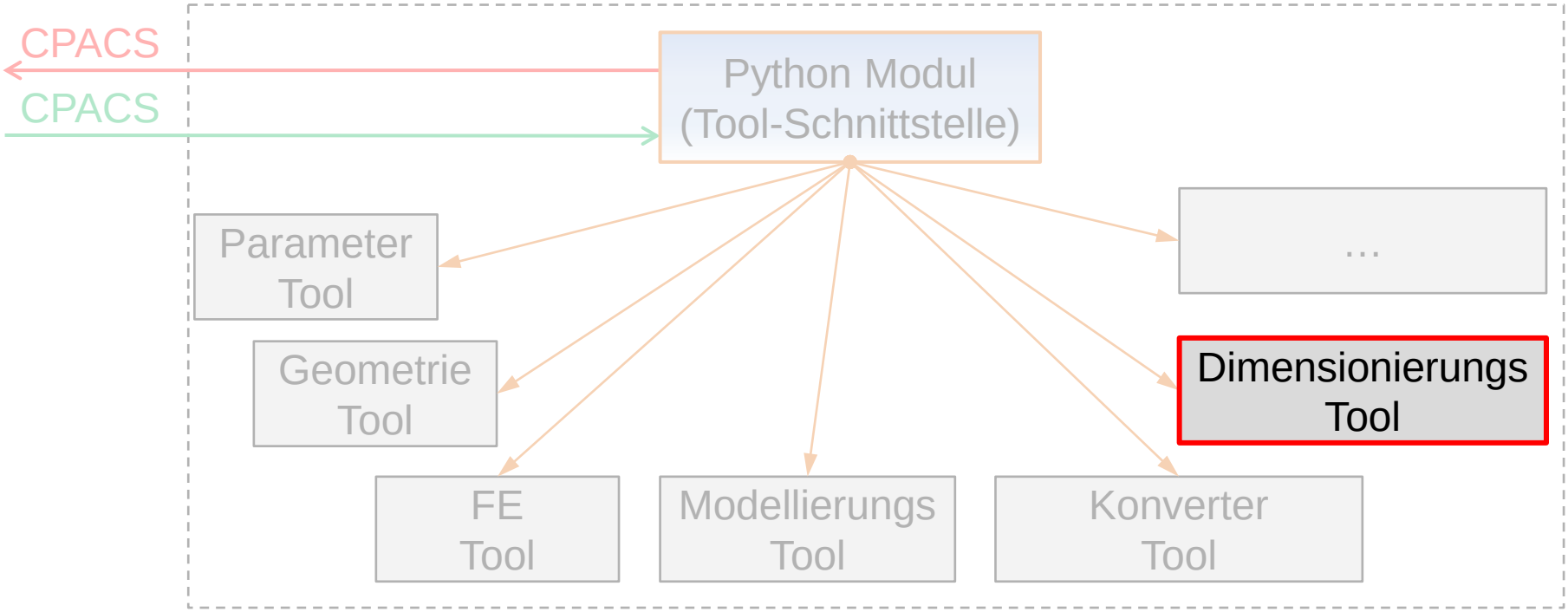
Anwenderschnittstellen

GUI und Batch-Betrieb möglich




```
%CONSPYRE_DIR%>python main.py --help
usage: main.py [-h] [-v] [-b]
               [-c "config file path" | -f "input path" "output
path"]

optional arguments:
  -h, --help                        show this help message and exit
  -v, --version                     View version information
  -b, --batch                       Run in batch mode.
  -c "config file path", --config "config file path"
                                    Specify a non-default config file
  -f "input path" "output path", --files "input path" "output path"
                                    Specify input and output paths
hidden argument:
  -s, --shell                       Show options for model shell cmds
```

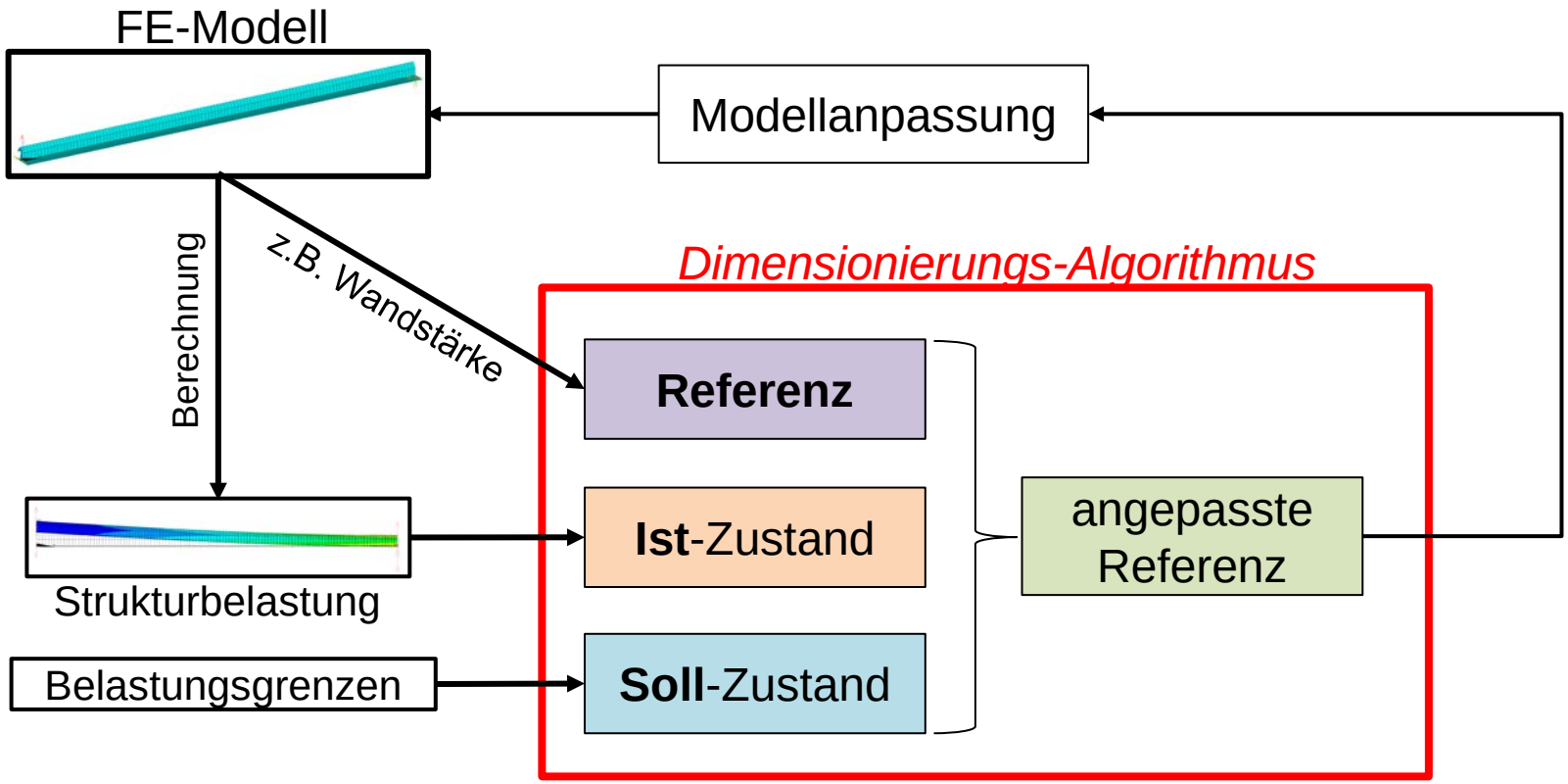
3) Open Source Toolumgebung zur Strukturauslegung



3) Open Source Toolumgebung zur Strukturauslegung

Dimensionierungs – Tool

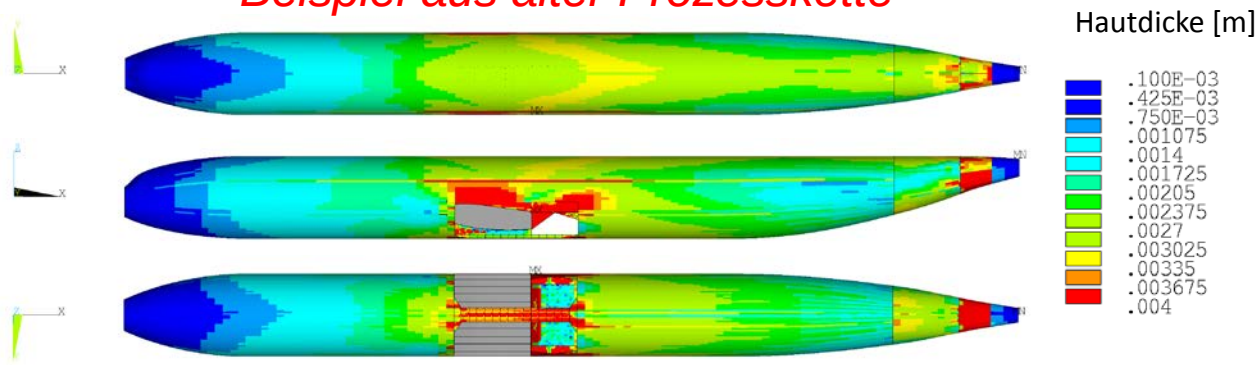
- Iterative Anpassung z.B. der Wandstärke
- Basierend auf Strukturbelastung



3) Open Source Toolumgebung zur Strukturauslegung

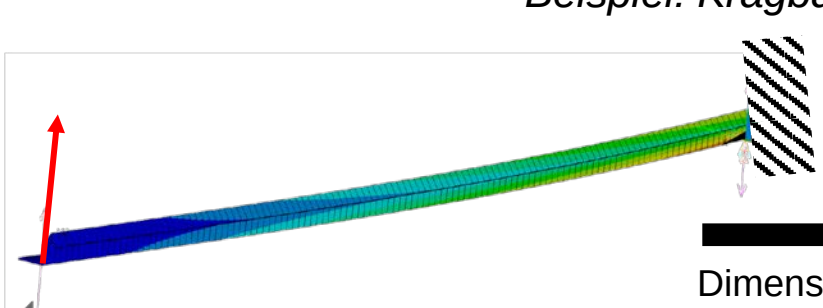
Dimensionierungs – Tool

Beispiel aus alter Prozesskette

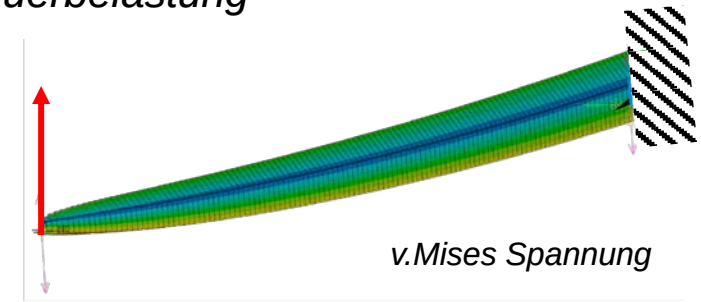


Erste Tests mit Python-Algorithmus und ANSYS-Solver

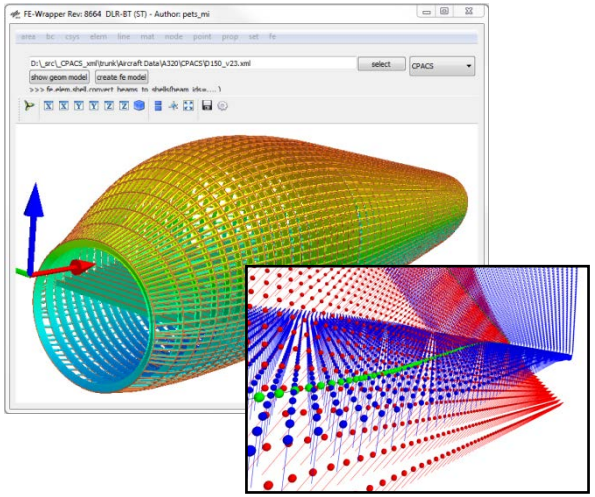
Beispiel: Kragbalken mit Querbelastung



Dimensionierung



4) Fazit

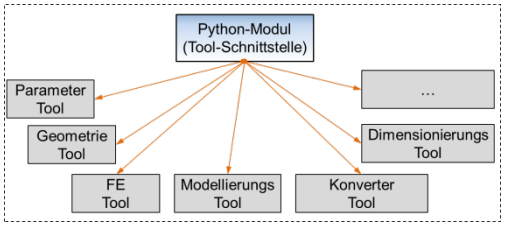


Einbindung vieler Module

- Schnelle Numerik (NumPy, SciPy, Pandas,…)
- Vielfältige Visualisierungstools (z.B. Mayavi, PyQt)

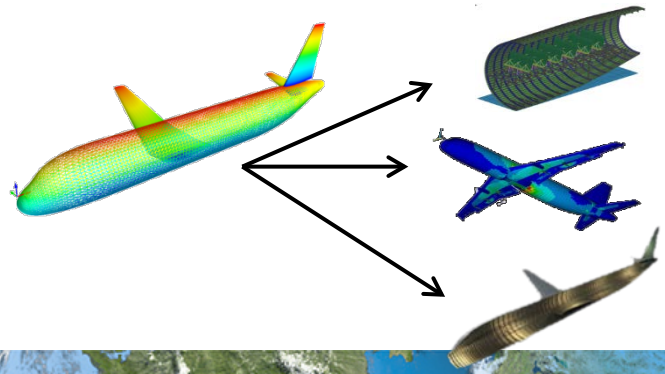
Modularisierung & Objektorientierung

- Klare Strukturen/Schnittstellen
- Trennung nach Funktionalität
- Einfachere Wart- und Erweiterbarkeit



Erweiterte Möglichkeiten

- Bereitstellung von Geometrie-Objekten
- Modularer Aufbau des FE-Modells
- Umsetzung spezifischer Funktionalitäten
- Implementierung von GUIs
- Exportieren verschiedener Formate



Michael Petsch (*michael.petsch@dlr.de*)
Deutsches Zentrum für Luft- und Raumfahrt (DLR)
Institut für Bauweisen und Strukturtechnologie (BT)
Pfaffenwaldring 38-40 | 70569 Stuttgart
Tel.: +49 711 6862 368

Danke für Ihre Aufmerksamkeit!