Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Fachbereich
Informatik

FrOSCon
Free and Open Source Software Conference

Science Track FrosCon 2016

von

Free and Open Source Software Conference

Veröffentlicht : 21.08.2016

Automatisierte Strukturauslegung im Flugzeugvorentwurf mit Python

Autor:
Michael
Petsch

1 Flugzeugvorentwurf Allgemein

Beim Entwurf eines effizienten und sicheren Luftfahrzeugs müssen viele fachliche Aspekte berücksichtigt werden. Die Bereiche Aerodynamik, Strukturmechanik als auch Flugmechanik spielen eine wichtige Rolle und hängen voneinander ab. Daher ist ein iterativer Entwurfsprozess erforderlich, um einen an die Anforderungen bestmöglich angepassten Kompromiss zu finden. In der Forschung am Deutschen Zentrum für Luft- und Raumfahrt e.V. (DLR) werden dafür automatisierte Prozessketten entwickelt, die zur Bewertung und Entwicklung von neuen Flugzeugkonzepten dienen. Das Institut für Bauweisen und Strukturtechnologie (BT) des DLR in Stuttgart hat dafür das Rumpfstrukturmodellierungs und -dimensionierungs-Tool (TRAFUMO [1][2]) entwickelt. Um mehr Schnittstellen mit anderen Tools zu schaffen, wird aktuell eine Python basierte Open Source Toolumgebung als Nachfolger von TRAFUMO entwickelt, welche im Kapitel 3 dargestellt wird.

2 Prozesskette Flugzeugvorentwurf am DLR

Für den automatisierten, multidisziplinären Entwurfsprozess eines Luftfahrzeugs am DLR werden spezifische Vorentwurfs-Tools über die Remote Component Environment (RCE) [3] Software des DLR zu einer Prozesskette verbunden. Um den Datenaustausch zwischen verschiedenen Vorentwurfs-Tools zu realisieren wurde ein XML basiertes Datenaustauschformat entwickelt. Dieses Austauschformat genannt Common Aircraft Configuration Schema (CPACS [4][5]) wird fortlaufend erweitert und enthält Parameter zur Beschreibung eines Luftfahrzeugentwurfs. Diese reichen von der Beschreibung der Oberflächengeometrie über die Definition von Strukturkomponenten bis hin zu Missionsbeschreibungen.
Das Tool TRAFUMO vom DLR-BT basiert weitestgehend auf der Skriptsprache APDL der kommerziellen Finite Elemente (FE) Software ANSYS [6]. Neben den Vorteilen mit APDL auf viele Funktionalitäten in ANSYS skriptbasiert zugreifen zu können, ergeben sich gleichzeitig viele Nachteile z.B. durch die eingeschränkte Nutzung von Vektoroperationen gegenüber einer universellen Programmiersprache wie Python mit umfangreichen Modulen wie NumPy. Die fehlende Objektorientierung erschwert den Aufbau einer übersichtlichen und modularen Toolumgebung. Auch lässt sich der Funktionsumfang in APDL nur bedingt erweitern, wodurch komplexere Algorithmen nicht realisierbar sind oder aufwändig implementiert werden müssen und dadurch mehr Rechenzeit benötigen als vergleichbare Umsetzungen in Python.
Um die Strukturmodellierungs- und Optimierungsmöglichkeiten zukünftig offener und flexibler zu gestalten, wird aktuell eine komplett Python basierte Toolumgebung entwickelt. Erweiter-

te Möglichkeiten ergeben sich durch die Einbindung von Modulen wie NumPy, Pandas oder Mayavi sowie in der Nutzung von weiterer Open Source Software. Ein Einblick in den Aufbau sowie die Möglichkeiten der Open Source basierten Toolumgebung wird im Kapitel 3 gegeben.

3 Open Source Toolumgebung zur Strukturauslegung

Ziel der Open Source basierten Toolumgebung für die Strukturauslegung ist sowohl die Generierung einer Luftfahrtstruktur (primär Rumpfstruktur) über einen CPACS-Datensatz sowie die iterative Anpassung der Struktur an auftretende Lasten. Eine Verwendung des Strukturmodells für zusätzliche Crash-Berechnungen stellt zudem ein Sekundärziel dar. Die klar definierten Aufgabenbereiche dieser Toolumgebung repräsentieren auch die einzelnen Tools, diese sind:

- Verwaltung von CPACS XML-Daten (Kapitel 3.1)
- Aufbereitung von CPACS-Geometrie (Kapitel 3.2)
- Verwaltung von FE-Daten (Kapitel 3.3)
- Modellierung eines Basis-FE-Modells (Kapitel 3.4)
- Modellierung eines detaillierten FE-Modells (Kapitel 3.5)
- Konvertierung von FE-Daten (Kapitel 3.6)
- Dimensionierung von FE-Daten (Kapitel 3.7)

Durch diese Trennung vereinfacht sich die Ausbaufähigkeit der einzelnen Tools und gleichzeitig können weitere Anwendungsgebiete erschlossen werden. Die ausschließliche Verwendung von Open Source Software ermöglicht auch den Austausch mit einer größeren Anzahl an Entwickler und Nutzer. Die Erstellung eines FE-Strukturmodells aus einem CPACS-Datensatz basiert auf dem Grundgedanken, zuerst ein einfaches Basismodell zu generieren, welches bei allen Modellierungszielen (Dimensionierung, Crash,...) gleich ist. Ausgehend von diesem FE-Solver unabhängigen Basismodell kann die Komplexität bei Bedarf modular gesteigert werden (z.B. durch spezifische Modellierungen und Netzanpassungen), um ein detallierteres FE-Modell (z.B. für Crashberechnungen) zu erhalten. Eine kurze Beschreibung der in Entwicklung befindlichen Tools folgt in den Kapiteln 3.1 bis 3.7.

3.1 Verwaltung von CPACS-XML-Daten

Zum Verwalten des CPACS-Datensatzes, und damit aller Parameter zur Beschreibung des Luftfahrzeugentwurfs, wurde ein objektorientiertes Modul entwickelt (basierend auf dem Python Modul lxml). Neben Möglichkeiten zum Modifizieren, Validieren und Iterieren von CPACS-Daten sind zusätzliche Funktionalitäten integriert, um die Nutzung zu vereinfachen:

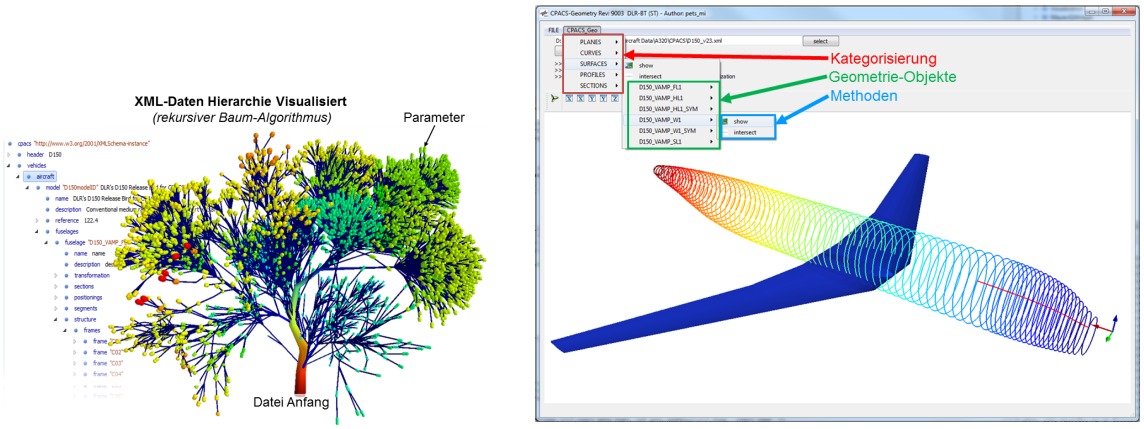
- Automatische Konvertierung von XML-Text zu Skalar- oder Vektordaten.
- Autovervollständigung eines Pfades in der Konsole.
- Setzen von Links zwischen verknüpften Objekten.
- Konvertierung von einheitlichen Unterästen in Tabellenform.

- Visualisieren der Datenstruktur.

Die Visualisierung von CPACS-XML-Daten (siehe Abbildung 3a) erfolgt über einen einfachen Baumalgorithmus. Jeder Verzweigung (Ast) des Datensatzes wird rekursiv eine Punkt und Vektor zugewiesen, basierend auf dem übergeordneten Ast, der Anzahl Unteräste sowie weiteren Parametern. Diese Art der Visualisierung kann z.B. zum Hervorheben von Änderungen in CPACS genutzt werden.

3.2 Aufbereitung von CPACS-Geometrie

Die Oberflächengeometrie des Luftfahrzeugs ist im CPACS-Datensatz nicht direkt in Form von Flächendefinitionen gegeben sondern wird z.B. über die Interpolation zwischen Stützprofilen beschrieben. Durch weitere Definitionen wie z.B. Schnittebenen mit der Oberfläche werden Referenzen für Strukturkomponenten definiert. Durch diese indirekte Definition der Geometrie lässt sich die Beschreibung des Luftfahrzeugs leichter modifizieren, ist aber auch aufwändiger zu extrahieren und aufzubereiten. Ein bereits am DLR entwickeltes und frei verfügbares Geometrie-Tool TIGL [7], basierend auf dem Geometrie-Kern OpenCascade (OCC) [8], bietet die Möglichkeit die Oberflächengeometrie darzustellen sowie einfache Verschneidungen durchzuführen.
Um die Laufzeit der Toolumgebung bei vielen Geometrieoperationen zu reduzieren, werden aktuell zusätzliche Möglichkeiten zur Geometrienutzung am DLR Institut BT untersucht. So kann durch die Erstellung der Geometrie direkt in Python über pythonOCC [9] der Overhead reduziert werden, während voller Zugriff auf alle OCC-Geometrieoperationen besteht. Eine weitere Möglichkeit besteht in der Nutzung von MESH-Geometrie (diskretes Netz aus Punkten), wofür in Python eigene Algorithmen (basierend auf Modulen wie NumPy, Pandas oder SciPy) entwickelt wurden. Bei variabler Netzgröße der Geometrie kann so die Rechenzeit und Genauigkeit entsprechend den Anforderungen variiert werden. So sind z.B bei aufwändigen Verschnitten Reduktionen der Rechenzeit um Größenordnungen möglich, was bei der Berechnung vieler Geometriekomponente von Vorteil sein kann. Steht dagegen die Geometriequalität (Genauigkeit, Gradienten) im Vordergrund ist die OCC-Geometrie zu bevorzugen. Ein objek-



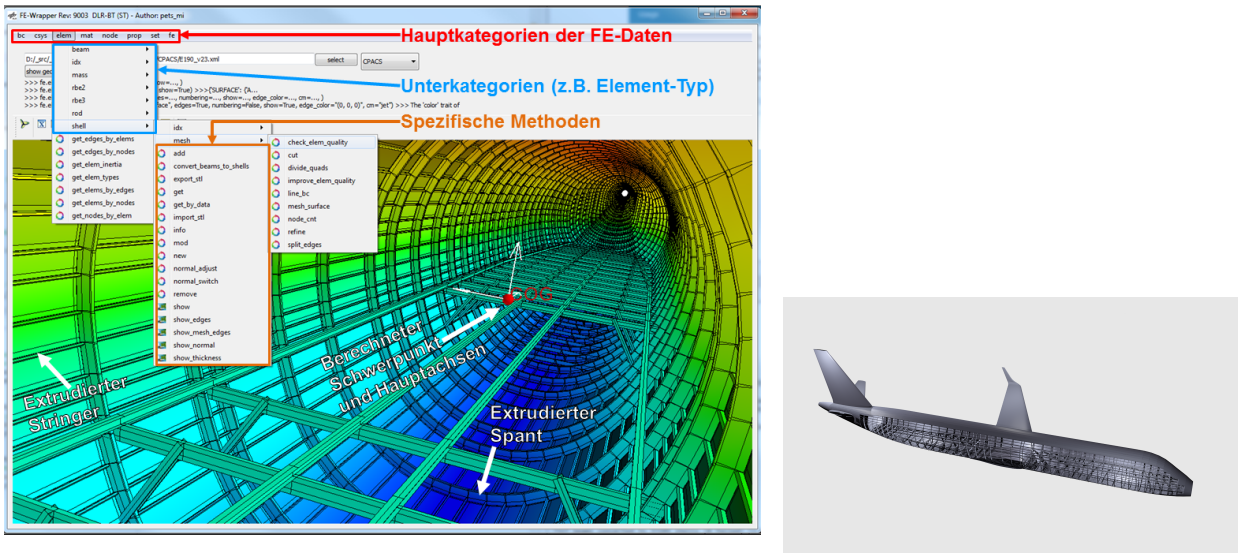
(a) CPACS Hierarchie Visualisierung. **(b)** CPACS Geometrie-Modul GUI.

Abbildung 3: Visualisierung von CPACS Daten und Geometrie.

torientiertes Geometrie-Modul zur Verwaltung der in CPACS beschriebenen Geometrie bietet hier die Möglichkeit entweder MESH- oder auch OCC-Geometrie zu nutzen, indem jede Geometrie unabhängig vom Geometrie-Kern und -Typ als einheitliches Geometrie-Objekt mit unterschiedlichen Eigenschaften definiert ist. Geometrieoperationen wie z.B. Verschnitte können so zwischen beliebigen Geometrie-Objekten über eine einzige Methode ausgeführt werden, wobei die benötigten Geometrie-Operationen abhängig vom Geometrie-Kern und -Typ automatisch aufgerufen werden und ein neues Geometrie-Objekt zurückgeliefert wird. In Abbildung 3b ist beispielhaft eine GUI des Geometrie-Moduls gezeigt, mit den Möglichkeiten der Kategorisierung der aus CPACS aufbereiteten und zur Verfügung gestellten Geometrie-Objekten.

3.3 Verwaltung von FE-Daten

Um einen flexiblen und FE-Solver unabhängigen FE-Modellaufbau zu ermöglichen, wurde ein Tool zur Verwaltung von FE-Daten (FE-PreProzessor) entwickelt. Neben dem Modifizieren von Knoten, Elementen oder Randbedingungen sind auch spezifische Funktionalitäten wie die Konvertierung von vereinfachten Balken- in Schalenelemente implementiert. Weiterhin bestehen Möglichkeiten für verschiedene Visualisierungen (z.B. Elementnormalen), Netzmodifikationen oder Modell-Exporte (z.B. für FE-Solver über das Konverter-Tool oder als STL-Format für CAD- und Rendering-Software). Als Kern des FE-PreProzessors wird das interne FE-Daten-Format des Konverter-Tools (Kapitel 3.6) genutzt. Neben der skriptbasierten Nutzung des FE-PreProzessors über Python ist auch eine mit PyQt erstellte GUI (siehe Abbildung 3a) implementiert, um den Aufbau des Tools zu Visualisieren und einem Anwender die Möglichkeit zu geben ohne Programmierung ein FE-Modell zu erstellen.



(a) FE-PreProzessor GUI. **(b)** Gerendertes Basis-FE-Modell.

Abbildung 3: FE-PreProzessor und Basis-FE-Modell.

3.4 Modellierung eines Basis-FE-Modells

Das Basis-FE-Modell kann sowohl basierend auf der OCC- als auch der MESH-Geometrie (siehe Kapitel 3.2) aufgebaut werden. Über die Oberflächengeometrie können je nach benötigter Netzfeinheit sowohl einfachere Verschnittpunkte als auch aufwändigere Verschnittkurven berechnet werden, um die Rumpfstruktur des FE-Basismodells aufzubauen. Dazu zählen luftfahrttypische Strukturkomponenten wie Spante (umlaufende Versteifungen) oder Stringer (Längsversteifungen), welche in Abbildung 3a eingezeichnet sind. Des Weiteren werden über CPACS die FE-Eigenschaften wie Materialien oder Hautdicken definiert sowie realitätsnahe Modellrandbedingungen festgelegt. Spezifische Anpassungen des Basismodells an eine reale Luftfahrtstruktur werden über erweiterte Modellierungs-Tools (Kapitel 3.5) realisiert. In Abbildung 3b ist beispielhaft ein gerendertes Basis-FE-Modell mit der freien GPL-lizensierten 3D-Grafiksuite Blender [10] dargestellt.

3.5 Modellierung eines detaillierten FE-Modells

Zur erweiterten Ausmodellierung des Basis-FE-Modells zählen z.B. Fahrwerksschächte, Flügelanbindungen oder Druckschotte. Diese Komponenten werden basierend auf dem Basis-FE-Modell über spezifische Definitionen in CPACS sowie angepasste Algorithmen generiert. Für Ausschnitte, Netzverfeinerungen oder Extrusionen können Methoden des FE-PreProzessors verwendet werden. Das detaillierte FE-Modell kann dann für die Strukturdimensionierung oder durch weitere Anpassungen (z.B. der Netzfeinheit) auch für Crashsimulationen verwendet werden. Eine Umsetzung dieser bisher weitesgehend APDL basierten Algorithmen in Python ist momentan in Entwicklung.

3.6 Konvertierung von FE-Daten

Der FE-Modell Konverter (Conspyre) ist zentraler Bestandteil der Open Source Toolumgebung und kann sowohl als eigenständiges Tool für die FE-Konvertierung zwischen zwei FE-Solver-Formaten verwendet werden, als auch um ein über die Toolumgebung erstelltes FE-Modell zu exportieren. Damit stellt Conspyre die Schnittstelle zwischen FE-Modellen und kommerziellen sowie Open Source FE-Solvern her und kann durch modulare sowie objektorientierte Programmierung für beliebige FE-Formate erweitert werden. Als Kern dienen Pandas-DataFrames welche gut zur Verwaltung großer Datenmengen geeignet sind. Genutzt werden kann Conspyre sowohl über die Kommandozeile sowie über eine GUI basierend auf PyQt. Der Konvertierungsablauf gliedert sich in drei Hauptebenen:

- Die Parser-Klasse (spezifisches Einlesen und Aufbereiten der FE-Daten).
- Die Model-Klasse (Verwalten und Visualisieren der FE-Daten im einheitlichen Format).
- Die Writer-Klasse (spezifisches Schreiben der eingelesenen FE-Daten).

Die spezifischen Parser- und Writer-Klassen bestehen dabei aus einem gemeinsamen Kern-Modul (enthält die Basisfunktionalitäten) dass über Vererbung um ein Plugin-Modul (enthält dem jeweiligen FE-Format angepasste Methoden) erweitert wird.

3.7 Dimensionierung von FE-Modellen

Für die Dimensionierung eines FE-Modelles (z.B. Anpassung der Wandstärke an vorgegebene Lasten) wird ein iterativer Prozess durchlaufen. Eine Iteration besteht dabei aus der Berechnung der Strukturbelastung des FE-Models (über einen FE-Solver), dem Dimensionierungsalgorithmus zur Berechnung der neuen Ausgangswerte (z.B. Hautdicken) sowie der Anpassung des FE-Modells (z.B. über den FE-PreProzessor, Kapitel 3.3).
Für die Berechnung können dabei mittelfristig beliebige FE-Solver verwendet werden. Die Konvertierung in das passende Format erfolgt über das Konverter-Tool (Kapitel 3.6). Für die Dimensionierung wurde bisher ein APDL-Skript basiertes Tool (S-BOT+ [1]) verwendet. Zukünftig wird dieses durch Python basierte Dimensionierungsalgorithmen abgelöst. Durch klare Trennung des Algorithmus von der Modellerstellung und -anpassung können auch weitere Optimierungsaufgaben ermöglicht werden. Erste Tests mit Python-Algorithmen zeigen bereits deutliche Laufzeitreduktionen gegenüber bisher verwendeter APDL-Skripte.

4 Fazit und Ausblick

Der komplette Umbau des Strukturmodellierungs- und Dimensionierungs-Tools der bisher weitestgehend APDL basierten Prozesskette hin zu einer Python basierten Toolumgebung zeigt bereits viele Vorteile durch den modularen und objektorientierten Aufbau. Aspekte der Wartung und Übersichtlichkeit vereinfachen sich so erheblich. Schnittstellen lassen sich leichter implementieren und durch die Einbindung zahlreicher Module (z.B. Numpy, SciPy, Pandas , Mayavi oder PyQt) reduzieren sich die Laufzeiten erheblich und ermöglichen auch die Implementierung von komplexeren Algorithmen sowie von GUIs. Durch das offen zugängliche FE-Daten-Format ergeben sich außerdem viele neue Funktionalitäten, welche z.B. über den FE-PreProzessor (Kapitel 3.3) oder den FE-Konverter (Kapitel 3.6) zugänglich sind. Weitere Herausforderungen stellen noch feinere Vernetzungen und Modellierungen dar, um auch detailliertere Crashmodelle generieren zu können. Dafür müssen zukünftig komplexere Vernetzungsalgorithmen sowie Geometriebeschreibungen implementiert werden.

Literaturverzeichnis

[1] J. Scherer, D. Kohlgrüber, F. Dorbath und M. Sorour. *Finite element based Tool Chain for Sizing of Transport Aircraft in the Preliminary Aircraft Design Phase*. Stuttgart (Germany): 62. DLRK, 2013 (zitiert auf Seiten 1, 6).

[2] J. Scherer und D. Kohlgrüber. *Overview of the versatile options to define fuselage structures within the CPACS data format*. Aachen (Germany): 4th EASN Workshop on Flight Physics & Aircraft Design, 2014 (zitiert auf Seite 1).

[3] DLR-SC. *Remote Component Environment*. 2016. URL: http://www.dlr.de/sc/desktopdefault.aspx/tabid-5625/9170_read-17513/ (besucht am 18. 08. 2016) (zitiert auf Seite 1).

[4] B. Nagel, D. Böhnke, V. Gollnick, P. Schmollgruber, A. Rizzi u. a. *Communication in Aircraft Design: Can We Establish a Common Language?* Brisbane (Australia): 28th Congress of the International Council of the Aeronautical Sciences (ICAS), 2012 (zitiert auf Seite 1).

[5] DLR-LY. *Common Parametric Aircraft Configuration Schema (CPACS)*. 2016. URL: https://github.com/DLR-LY/CPACS (besucht am 15. 08. 2016) (zitiert auf Seite 1).

[6] Ansys Inc. *ANSYS FE-Software*. 2016. URL: http://www.ansys.com/ (besucht am 15. 08. 2016) (zitiert auf Seite 1).

[7] DLR-SC. *The TiGL Geometry Library to process aircraft geometries in pre-design.* 2016. URL: https://github.com/DLR-SC/tigl (besucht am 15. 08. 2016) (zitiert auf Seite 3).

[8] Open Cascade S.A.S. *Open CASCADE Technology*. 2016. URL: https://www.opencascade.com (besucht am 15. 08. 2016) (zitiert auf Seite 3).

[9] Open Cascade S.A.S. *Open Cascade API für Python*. 2016. URL: http://www.pythonocc.org/ (besucht am 15. 08. 2016) (zitiert auf Seite 3).

[10] Blender Foundation. *Blender 3D-Grafiksuite*. 2016. URL: https://www.blender.org/ (besucht am 18. 08. 2016) (zitiert auf Seite 5).