

Diplomarbeit

Entwicklung eines Geometrie-Generators
zur Initialisierung von unkonventionellen
Flugzeugentwürfen in CPACS

Bearbeiter: Jonas Jepsen
Matrikelnummer: 31980
Studiengang: Flugzeugsystemtechnik
Erstprüfer: Prof. Dr.-Ing. D. Krause
Zweitprüfer: Prof. Dr.-Ing. V. Gollnick
Betreuer: Dipl.-Ing. Daniel Böhnke

Hamburg, 13.06.2012

**Aufgabenstellung der Diplomarbeit von
Herrn Jonas Jepsen, Matr.-Nr. 31980**

Thema:

**Entwicklung eines Geometrie-Generators zur Initialisierung
von unkonventionellen Flugzeugentwürfen in CPACS**

Die aktuellen Herausforderungen der Luftfahrt in Hinsicht auf eine versiegende Versorgung mit fossilen Brennstoffen, Umweltbelastung und ein steigendes Passagieraufkommen, verlangen den Entwurf neuartiger Konfigurationen. Es ist abzusehen, dass mit einer kontinuierlichen Weiterentwicklung existierender Technologien angestrebte Ziele für den Luftverkehr der Zukunft nicht erreicht werden können. Ein Technologiesprung würde sich mit der Einführungen unkonventioneller Konfigurationen etablieren lassen.

Um verlässliche Aussagen über unkonventionelle Konfigurationen im Entwurfprozess treffen zu können ist die Anwendung von Verfahren auf physikalischer Basis zwingend notwendig. Nur durch diese Verfahren ist es möglich die steigenden Anzahl von Abhängigkeiten im System Flugzeug abzubilden.

Eine offene Entwurfsumgebung für den Flugzeugentwurf entsteht zur Zeit in Deutschland. Zentraler Bestandteil der in dieser Umgebung aufgebauten Entwurfsprozesse ist ein zentrales Datenformat CPACS (Common Parametric Aircraft Configuration Schema). Es ist nunmehr möglich sowohl eine Vielzahl von Konfigurationen (Conventional, BWB, Box-Wing) als auch verschiedene Detailstufen (Vortex-Lattice, Euler, RANS) zu analysieren.

Ziel der ausgeschriebenen Diplomarbeit ist es ein Programm zu entwickeln, dass es ermöglicht die notwendigen Daten für eine Initialisierung der Entwurfsprozesse zur Verfügung zu stellen. Anforderungen an das Programm leiten sich zum einen ab aus der Absicht unkonventionelle Konfigurationen zu bearbeiten. In einem einheitlichen Ansatz soll eine parametrische Möglichkeit definiert werden, die es ermöglicht in einem möglichst großen Entwurfsraum Konfigurationen zu entwickeln. Zum Anderen soll am Beispiel einer Entwurfskette nachgewiesen werden, dass Analysewerkzeuge im späteren Entwurfsprozess angesprochen werden können.

Verwandte Arbeiten sind bisher an der TU Delft mit der Design Engineering Engine und dem Vehicle Sketch Pad der NASA entstanden. Die ausgeschriebene Arbeit hebt sich von diesen durch die Fokussierung auf ein zentrales Datenformat ab.

Ausgabe der Aufgabenstellung am: 16. Februar 2012

Abgabe der Arbeit bis: 15. Juni 2012

Betreuer der Arbeit: Prof. Dr.-Ing. D. Krause
Dipl.-Ing. D. Böhnke

Prof. Dr.-Ing. D. Krause Jonas Jepsen

Eidesstattliche Erklärung

Ich, JONAS JEPSEN (Student der Flugzeugsystemtechnik an der Technischen Universität Hamburg-Harburg, Matrikelnummer 31980), versichere an Eides Statt, dass ich die vorliegende Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Die Arbeit wurde in dieser oder ähnlicher Form noch keiner Prüfungskommission vorgelegt.

Hamburg, 13.06.2012

Jonas Jepsen

Übersicht

Das prognostizierte Wachstum des Luftfahrtaufkommens sowie wachsende ökonomische und ökologische Anforderungen erfordern starke Maßnahmen im Bereich des Flugzeugentwurfs. Um zukunfts-fähig zu bleiben, sind deutliche Verbesserungen der Flugzeugentwürfe notwendig. Untersuchungen von unkonventionellen Entwürfen, wie z.B. dem Blended Wing Body liefern vielversprechende Ergebnisse. Moderne Entwürfe setzen, im Gegensatz zu der Funktionsdifferenzierung traditioneller Entwürfe, immer stärker auf Funktionsintegration. Aufgrund fehlender Daten, können statistische Handbuchmethoden für innovative Entwürfe nur bedingt eingesetzt werden. Daher wird daran gearbeitet physikalische Analyseverfahren, sowie eine Multi-Disziplinäre-Optimierung (MDO) bereits in die frühen Phasen des Entwurfsprozesses zu integrieren.

In dieser Arbeit wird die Entwicklung eines Softwaretools, dem Simple Geometry Generator (SGG), beschrieben, dass zum einfachen Generieren von geometriebasierten Flugzeugentwürfen dient. Die erstellten Geometrien werden in die vom DLR entwickelte CPACS-Schnittstelle (Common Parametric Aircraft Configuration Schema) exportiert und können so als Ausgangspunkt einer MDO verwendet werden. Das entwickelte Programm bietet die Modellierung und Bearbeitung von Flugzeuggeometrien anhand von intuitiven Entwurfsparametern über ein grafisches Benutzerinterface. Als Entwurfsparameter stehen unter Anderem die Pfeilung, V-Stellung, Spannweite, Zusitzung, Streckung und Referenzfläche zur Verfügung, die in verschiedenen Kombinationen verwendet werden können. Durch die Verwendung von Teilstücken (Parts), werden die Entwurfsparameter von den CPACS-Parametern getrennt und der Umgang mit den Geometrien vereinfacht. Für die Verifizierung der Geometrien dienen drei Ansichten, in denen die Auswirkungen von Parameteränderungen live beobachtet werden können. Sowohl die Erstellung von Flugzeugentwürfen, als auch die Verwendung der erstellten Geometrien in einem Analysetool wurden bereits erfolgreich getestet und waren einfach durchzuführen.

Inhaltsverzeichnis

Abkürzungsverzeichnis	VI
Formelzeichen	VIII
1 Einleitung	1
2 Entwicklung des Flugzeugentwurfsprozesses	3
2.1 Entwicklung der Luftfahrt	3
2.2 Traditioneller Flugzeugentwurf.	4
2.2.1 Konzeptentwurf	5
2.2.2 Vorentwurf.	5
2.2.3 Detaillentwurf	6
2.3 Ganzheitlicher Flugzeugentwurfsprozess	8
2.3.1 MDO	10
2.3.2 Entwurfsumgebungen.	11
2.3.3 Zentrale Schnittstelle	14
2.3.4 Analysetools	17
2.3.5 Initialisierung	18
2.4 Zusammenfassung/Fazit	20
3 Simple Geometry Generator (SGG)	22
3.1 Anforderungen an die Software.	22
3.2 CPACS	27
3.2.1 CPACS-Pfad.	30
3.3 Umgang mit Geometrien.	31
3.3.1 Parts	31
3.3.2 Koordinatenberechnung.	38
3.3.3 Erstellen von Geometrien	42
3.3.4 Entfernen/Löschen von Komponenten	43
3.4 Aufbau/Struktur.	44
3.4.1 Grafisches Layout	44
3.4.2 Datenmodell.	48

3.4.3 Programmlogik	52
3.5 Grafische Benutzerschnittstelle	53
3.5.1 Menü.	54
3.5.2 Baumansicht.	54
3.5.3 TransformationWidget	54
3.5.4 AircraftWidget	55
3.5.5 ComponentWidget	56
3.5.6 WingPartWidget	58
3.5.7 FuselagePartWidget	60
3.5.8 SectionWidget	62
3.5.9 PositioningWidget	62
3.5.10 SegmentsWidget	63
3.5.11 SegmentWidget	63
3.5.12 ElementWidget	64
3.5.13 3 Seitenansichten	64
3.5.14 Einstellungen	65
3.6 TIGLViewer.	67
4 Testen des SGG	69
4.1 Erstellen von Flugzeugentwürfen	69
4.1.1 Drachenkonfiguration	69
4.1.2 Boxwing	78
4.1.3 Blended-Wing-Body	82
4.2 Anbindung an Tornado.	86
4.3 Ergebnisse	88
5 Zusammenfassung und Ausblick	92
Literaturverzeichnis	94
Anhang	97
A Anhang	97
A.1 generateDS	97
A.2 Dateistruktur des SGG	97
A.3 Zusätzliche Features	98
A.4 Listings.	99
A.5 Tabellen	100

A.6 Anforderungsliste	101
---------------------------------	-----

Abkürzungsverzeichnis

Abk.	Bedeutung
ACARE	= Advisory Council for Aeronautics Research in Europe
API	= Application Programming Interface
ASCII	= American Standard Code for Information Interchange
BWB	= Blended Wing Body
CAD	= Computer Aided Design
CAM	= Computer Aided Manufacturing
CATS	= Climate Compatible Air Transportation System
CFD	= Computational Fluid Dynamics
CPACS	= Common Parametric Aircraft Configuration Schema
DEE	= Design and Engineering Engine
DLR	= Deutsches Zentrum für Luft- und Raumfahrt
DOC	= Direct Operating Costs
EVITA	= Evaluierung innovativer Turboantriebe
FEM	= Finite-Elemente-Methode
GUI	= Graphical User Interface
IATA	= International Air Transport Association
IGES	= Initial Graphics Exchange Specification
ISO	= International Organization for Standardization
KBE	= Knowledge Based Engineering
KOS	= Koordinatensystem
MDO	= Multi-Disziplinäre-Optimierung
MMG	= Multi-Model-Generator
OMG	= Object-Management-Group
PrADO	= Preliminary Aircraft Design and Optimization
RCE	= Remote Component Environment
SGG	= Simple Geometry Generator
STEP	= Standard for the Exchange of Product model data

Abk.	Bedeutung
TIGL	= TIVA Geometry Library
TIVA	= Technology Integration for the Virtual Aircraft
UCAV	= Unmanned Combat Air Vehicle
UID/uID	= Unique Identifier
UML	= Unified Modeling Language
VAMP	= Virtual Aircraft Multidisciplinary Analysis and Design Processes
VSP	= Vehicle Sketch Pad
XML	= Extensible Markup Language

Formelzeichen

Symbol	Einheit	Beschreibung
B	m	Querschnittsbreite
C_A	—	Auftriebsbeiwert
C_M	—	Momentenbeiwert
C_W	—	Widerstandsbeiwert
C_{Wi}	—	Beiwert für den induzierten Widerstand
C_{W0}	—	Nullwiderstandsbeiwert
H	m	Querschnittshöhe
L	m	Länge eines Teils
Ma	—	Machzahl
\vec{P}	m	Punktvektor
\vec{R}	rad	Rotationsmatrix
Re	—	Reynoldszahl
S_{ref}	m^2	Referenzfläche
\vec{S}	—	Skalierungsmatrix
\vec{T}	m	Translationsvektor
c	$\frac{m}{s}$	Schallgeschwindigkeit
d	m	Profildicke
e	—	Oswaldfaktor
$scale$	$\frac{px}{m}$	Maßstab
$span$	m	Spannweite
t	m	Profiltiefe
t_μ	m	Bezugsflügeltiefe
v	$\frac{m}{s}$	Strömungsgeschwindigkeit
w_{image}	px	Bildbreite
x	m	x-Komponente

y	m	y-Komponente
z	m	z-Komponente
ε	\circ	Geometrische Profilverwindung
η	—	dimensionslose Spannweitenkoordinate
ν	$\frac{m^2}{s}$	kinematische Viskosität
τ	—	Zuspitzung
φ	\circ	Pfeilung
ψ	\circ	V-Stellung
Λ	—	Streckung

Indizes

a	außen
elem	„element“-Knoten
from	„from“-Querschnitt
front	Ansicht von Vorne
i	innen
max	Maximum
min	Minimum
part	Part
P[i]	[i]-ter Part einer Komponente
sec	„section“-Knoten
side	Ansicht von der Seite
to	„to“-Querschnitt
top	Ansicht von Oben
trans	Translation
*	alter/vorheriger Wert
^	Kennzeichnet dimensionslose Skalierungsfaktoren

1 Einleitung

Der moderne Flugzeugentwurf ist ein sehr komplexer Prozess, an den ständig wachsende Anforderungen sowohl ökonomisch, als auch ökologisch gestellt werden. Diese Anforderungen umfassen z.B. gesteckte Ziele für die Zukunft der Luftfahrt wie sie von ACARE (Advisory Council for Aeronautics Research in Europe) [Adv01] und IATA (International Air Transport Association) [Int11] beschrieben werden. Dazu gehört z.B. die Senkung der Emission von Treibhausgasen, sowie eine Lärmminderung. Zusätzlich zu den Zielen, wird außerdem ein starker Anstieg des Luftverkehrsaufkommens prognostiziert.

Um die angestrebten Ziele zu erreichen ist ein sogenannter Technologiesprung erforderlich. Eine Möglichkeit für einen Technologiesprung ist das Umsetzen eines innovativen Flugzeugentwurfs. Der traditionelle Flugzeugentwurf weist zwei Mängel auf, die angegangen werden müssen um die Entwicklung innovativer Entwürfe zu fördern. Zum Einen verwendet der traditionelle Flugzeugentwurf viele statistische Verfahren, deren Gültigkeit sich auf Entwürfe beschränkt die den Entwürfen der verwendeten Statistik ähneln. Und zum Anderen werden zu Beginn des Entwurfs nur wenige Disziplinen berücksichtigt, was die wachsenden Anforderungen in anderen Disziplinen vernachlässigt. Daher ist ein Wandel des Entwurfsprozesses notwendig der zu einem ganzheitlichen Entwurfsprozess führt. Aufgrund dieser Mängel unterliegt der Flugzeugentwurfsprozess zur Zeit einem Wandel der zu einem ganzheitlichen Entwurfsprozess führt. Um zuverlässig auch unkonventionelle Flugzeugentwürfe erstellen zu können ermöglicht der ganzheitliche Entwurfsprozess die Verwendung physikalischer Vorentwurfsverfahren. Die vielfältigen Anforderungen aus unterschiedlichen Disziplinen erfordern außerdem, dass bereits frühzeitig alle relevanten Disziplinen berücksichtigt werden. Ein wesentlicher Bestandteil eines ganzheitlichen Entwurfsprozesses ist daher eine Multi-Disziplinäre-Optimierung (MDO). Für eine MDO wird ein initialer Flugzeugentwurf mittels Analysetools untersucht und anhand von Entwurfskriterien iterativ optimiert. Um eine MDO durchzuführen ist es notwendig die Entwürfe bereits frühzeitig anhand unterschiedlicher Analysetools zu untersuchen. Dies erfordert:

1. einen ausreichend hohen Detaillierungsgrad für die Analysetools.
2. passende Datenmodelle für die Analysetools.

Der höhere Detaillierungsgrad führt bereits in der Anfangsphase des Entwurfs zu größeren Datenmengen. Um diese bei angemessenem Aufwand verwalten zu können ist eine stärkere Automatisierung notwendig. Durch die Automatisierung repetitiver, nicht-kreativer Aufgaben können Flugzeugentwerfer zunehmend entlastet werden. Diese können sich

dann stärker richtungsweisenden Entwurfsaufgaben widmen.

Um eine MDO umzusetzen, sollte eine Entwurfsumgebung geschaffen werden, in der Entwurfswerkzeuge und Analysetools automatisiert miteinander Interagieren. Der Datenaustausch zweier Tools wird über deren Schnittstellen durchgeführt. Wenn für jedes Tool eine eigene Schnittstelle definiert ist, sind für den Datenaustausch Konvertierungsprogramme für beide Richtungen zu implementieren. Um ein Tool in ein System zu integrieren müssen Verbindungen zu jedem anderen Programm implementiert werden, mit dem das Tool interagieren soll. In einem System, dass aus vielen verschiedenen Tools besteht führt dies schnell zu einem erheblichen Aufwand. Eine zentrale Schnittstelle hilft dabei diesen Aufwand zu reduzieren indem neue Tools lediglich an die zentrale Schnittstelle angeschlossen werden müssen. Die Verbindung zu den anderen Tools im System ist dann über diese Schnittstelle möglich. Am Deutschen Zentrum für Luft- und Raumfahrt (DLR) wird daher seit 2005 eine zentrale Schnittstelle entwickelt, die Daten des gesamten Lebenszyklus eines Flugzeuges umfasst. Diese CPACS-Schnittstelle (Common Parametric Aircraft Configuration Schema) wird am DLR bereits in unterschiedlichen Softwaretools verwendet.

Um einen initialen Flugzeugentwurf für eine CPACS-basierte MDO zu erstellen, kann z.B. VAMPzero verwendet werden. VAMPzero ist eine Software für den Flugzeugkonzeptentwurf, die auf statistischen Methoden basiert. Der Entwurf innovativer Flugzeugentwürfe, die stark vom traditionellen Flugzeugentwurf abweichen, ist mit VAMPzero allerdings nur begrenzt möglich. Die Generierung eines unkonventionellen, geometriebasierten Entwurfs in CPACS ist bisher mit hohem Aufwand verbunden, da die CPACS-Struktur mit einem Text- oder XML-Editor von Hand erstellt werden muss. Der in dieser Arbeit entwickelte SGG (Simple Geometry Generator) ermöglicht eine einfache Generierung von Flugzeugentwürfen anhand von typischen, geometrischen Entwurfsparametern und ist dabei nicht auf konventionelle Entwürfe begrenzt.

Die vorliegende Arbeit untersucht aktuelle Entwicklungen im Bereich des konzeptionellen Flugzeugentwurfs und beschäftigt sich mit der Implementierung des „Simple Geometry Generators“. Um einen Überblick über aktuelle Entwicklungen des Flugzeugentwurfsprozesses zu bekommen, wird in Kapitel 2 zunächst die Entwicklung der Luftfahrt beschrieben und der traditionelle und ganzheitliche Entwurfsprozess vorgestellt. Dabei werden aktuelle Entwicklungen rund um den ganzheitlichen Flugzeugentwurf aufgegriffen.

Danach folgt in Kapitel 3 die Implementierung des Simple Geometry Generators, welche auch eine Beschreibung der verwendeten CPACS-Struktur beinhaltet. In Kapitel 4 wird die Funktion des SGG getestet und beispielhaft einige Flugzeugentwürfe erstellt. Um die Ausgaben (CPACS-Daten) des SGG zu testen, werden die erstellten Geometrien anhand von einem aerodynamischen Analyseprogramm (Tornado) untersucht.

Als letztes folgt in Kapitel 5 eine kurze Zusammenfassung der Arbeit und einige Vorschläge für die weitere Entwicklung des SGG.

2 Entwicklung des Flugzeugentwurfsprozesses

Um den Entwicklung des Flugzeugentwurfsprozesses besser zu verstehen ist es sinnvoll zunächst die Entwicklung der Luftfahrt zu betrachten. Daher wird in Abschnitt 2.1 als Erstes die Entwicklung der Luftfahrt beschrieben.

Um den im Allgemeinen sehr komplexen Flugzeugentwurf zu vereinfachen wurden mit wachsenden Erfahrungen geeignete Entwurfsprozesse entwickelt. Es wurden statistische Entwurfsverfahren entwickelt die den komplexen Flugzeugentwurf in sequenziell lösbar Teilaufgaben gegliedert haben. Die Sequenz, welche einfache physikalische, sowie statistische Gleichungen enthält stellt den ersten Teil des traditionellen Flugzeugentwurfsprozesses dar, der in Abschnitt 2.2 beschrieben wird. Dabei wird ein Flugzeugkonzeptentwurf auf Basis von typischen Anforderungen (wie Passagiere, Nutzlast, Reichweite) erstellt. Ständig wachsende Anforderungen (ökonomisch und ökologisch) machen es notwendig den Flugzeugentwurf weiter zu verbessern. Da die Grenzen heutiger Technologien beinahe erreicht sind, ist es notwendig einen Technologiesprung herbeizuführen. Der ganzheitliche Entwurfsprozess (Abschnitt 2.3), der die Defizite des traditionellen Entwurfsprozesses behebt ermöglicht eine bessere Entwicklung innovativer Flugzeugentwürfe.

2.1 Entwicklung der Luftfahrt

Die Entwicklung der Luftfahrt kann durch sogenannte S-Kurven beschrieben werden. Abbildung 2.1 zeigt die zwei großen S-Kurven seit Beginn der Luftfahrtgeschichte. Die

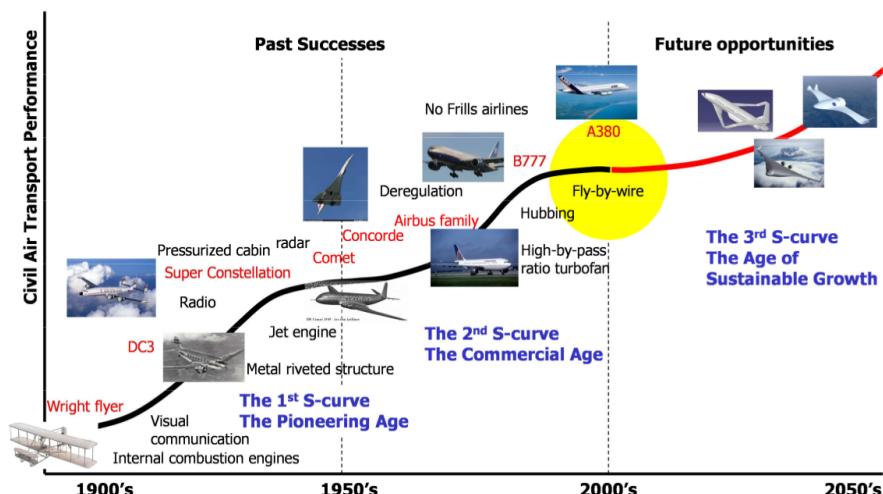


Abbildung 2.1: Entwicklung der Luftfahrt [La 11]

Abbildung zeigt wie sich die Transportleistung (Ordinate) im Verlauf der Jahre (Abszisse) verändert hat.

Am Anfang einer S-Kurve steht in der Regel eine vielversprechende Technologie mit großem Potential. Da mit der Einführung einer neuen Technologie bisher unbekannte Hindernisse überwindet werden müssen, beginnt die Kurve relativ flach. Durch den Umgang und die gesammelten Erfahrungen wird das Wissen über die Technologie erweitert und das Potential kann besser genutzt werden. Dies führt dazu, dass die Steigung der Kurve zunimmt. Je näher das erreichte Potential an das theoretische Maximum herankommt, desto aufwändiger werden weitere Verbesserungen. Wenn die Grenzen der neuen Technologien fast erreicht sind und langsam Sättigung eintritt, flacht die Kurve daher wieder stark ab. Durch weitere Optimierungen der bekannten Technologien kann die Transportleistung dann nur noch minimal verbessert werden. Abbildung 2.1, die nach [La 11] auf Daten von ACARE aus dem Jahre 2002 basiert, zeigt, dass sich die Entwicklung der Luftfahrt zur Zeit in einer Sättigungsphase befindet. Die Grenzen der heutigen Technologien sind fast erreicht, und um die Entwicklung weiter voran zu treiben sind neue Technologien erforderlich die erneut eine S-Kurve einleiten. Ein solcher Schritt wird im Allgemeinen als Technologiesprung bezeichnet.

In [Adv01] wird prognostiziert, dass das Luftfahrtaufkommen im Jahr 2020 bis zu drei mal so hoch ist wie noch im Jahr 2000. [Adv01] enthält außerdem viele Ziele wie zum Beispiel die Reduktion des CO_2 -Ausstoßes pro Passagier und Kilometer um 50%. In [Int11] werden entsprechende Prognosen für das Jahr 2050 getroffen. Diese von vom Advisory Council for Aeronautical Research in Europe (ACARE) und der International Air Transport Association (IATA) gesteckten Ziele können ohne einen Technologiesprung voraussichtlich nicht erreicht werden [La 11].

Um einen Technologiesprung herbeizuführen ist eine Entwicklung des Flugzeugentwurfsprozesses notwendig. Bereits seit einigen Jahren zeichnet sich ein Trend ab einen ganzheitlichen Flugzeugentwurf durchzuführen um Defizite des traditionellen Entwurfsprozesses zu beheben. Der traditionelle Entwurfsprozess und die Entwicklung zum ganzheitlichen Entwurfsprozess sind in den beiden folgenden Abschnitten beschrieben.

2.2 Traditioneller Flugzeugentwurf

Der traditionelle Flugzeugentwurf wird im Allgemeinen in drei Phasen unterteilt:

1. Konzeptentwurf
2. Vorentwurf
3. Detaillentwurf

Diese drei Phasen werden nacheinander durchlaufen. Innerhalb der einzelnen Phasen werden die Entwürfe durch mehrere Iterationsschritte laufend verbessert. Das Vorgehen in diesen drei Phasen ist zum Standard im Flugzeugentwurf geworden und wird auch

heute noch verwendet¹. Die Abläufe der einzelnen Phasen werden in den folgenden Abschnitten beschrieben.

2.2.1 Konzeptentwurf

Der Konzeptentwurf beginnt mit Entwurfsparametern, die in der Regel die Transportaufgabe beschreiben, die mit dem neuen Flugzeugentwurf erfüllt werden soll (Last X soll über Strecke Y transportiert werden). In dieser Phase werden grundlegende Fragestellungen bearbeitet. Im Vordergrund steht zunächst die Frage ob die Anforderungen plausibel sind. D.h. es wird die Frage beantwortet, ob es überhaupt einen Flugzeugentwurf gibt der die Anforderungen erfüllt. Falls dies nicht der Fall ist, müssen die Anforderungen überarbeitet werden.

Nach der Plausibilitätsprüfung werden unterschiedliche Flugzeugkonfigurationen auf ihre Tauglichkeit für die gestellte Transportaufgabe untersucht. Dabei werden möglichst viele unterschiedliche Flugzeugkonzepte betrachtet um dann die am besten geeigneten Konzepte/Konfigurationen herauszufiltern. Im Konzeptentwurf werden jedoch nur wenige Disziplinen zur Beurteilung der Güte unterschiedlicher Entwürfe herangezogen.

Diese Phase enthält den Entwurf der grundlegenden Geometrien von Tragflächen, Leitwerken und Rümpfen, sowie die Anordnung von Besatzung, Passagieren, Nutzlast, Equipment, Fahrwerken und Triebwerken [Ray99]. Die Geometrien sind dabei in der Regel wenig detailliert und werden anhand weniger Parameter beschrieben. Für die Tragfläche werden z.B. Parameter wie die Spannweite, die Flügelfläche, die Pfeilung und die V-Stellung verwendet. Entscheidend sind bei den Konfigurationen auch die Interaktionen zwischen den Komponenten wie z.B. am Flügel-Rumpf-Anschluss.

Die Dauer für die Konzeptentwurfsphase kann zwischen einer Woche (für einen dürftigen Entwurf) und mehreren Jahren liegen. Um jedoch die Alternativen ausreichend zu untersuchen sollten für den Konzeptentwurf in der Regel ca. 6 Monate eingeplant werden [Ray99].

Durch Computerunterstützung können die Abläufe im Konzeptentwurf automatisiert und Iterationen auf diese Weise schneller durchgeführt werden. Ein Beispiel für ein modernes Konzeptentwurfstool ist VAMPzero, welches in Abschnitt 2.3 näher erläutert wird.

2.2.2 Vorentwurf

Die Vorentwurfsphase beginnt in der Regeln nachdem einige Flugzeugkonfigurationen aus dem Konzeptentwurf ausgewählt wurden, um näher untersucht zu werden. Da die im Konzeptentwurf entstandenen Entwürfe sehr wenig detailliert sind, müssen diese vor der Untersuchung mittels Analysetools zunächst aufbereitet werden. So muss z.B. für eine CFD-Analyse die Flugzeuggeometrie in ein geeignetes Format überführt werden

¹ Teilweise in etwas abgewandelter Form, siehe Abschnitt 2.3

(Meshgenerierung). Dafür können CAD-Programme, oder speziell entwickelte Tools verwendet werden, die für den Flugzeugentwurf typischen Geometrieparameter in CFD taugliche Modelle überführen.

In einer Optimierungsschleife werden die Ergebnisse ausgewertet und die Entwürfe angepasst um ein Optimum bezüglich gewählter Kriterien zu erreichen. Falls trotz der Optimierung die Anforderungen nur unzureichend erfüllt werden, kann die Entscheidung getroffen werden noch einmal in den Konzeptentwurf zurück zu gehen.

Wenn der Entwurf der äußeren Geometrie zu einem zufriedenstellenden Ergebnis gelangt ist, wird der Entwurf eingefroren. Ab diesem Zeitpunkt kann mit dem Entwurf der inneren Struktur (z.B. Holme und Rippen) begonnen werden. Wenn die innere Struktur erstellt wurde kann in einer FEM Berechnung die Gesamtstruktur des Flugzeugentwurfs analysiert werden. Dafür müssen, wie bereits für die CFD-Analyse entsprechende Gitternetze erstellt werden. Dafür können ebenfalls geeignete CAD-Programme verwendet werden. Tools für eine FEM-Analyse gehören bereits zur Ausstattung einiger CAD-Programme wie z.B. Catia.

Auch Subsysteme wie z.B. die Fahrwerke werden erst nach dem Einfrieren der globalen Geometrie entworfen, da globale Änderungen einen großen Effekt auf die Subsysteme haben können. Während der Entwicklung der Subsysteme werden diese von Spezialisten mit Hilfe von Analysetools untersucht und verbessert.

Die Vorentwurfsphase endet in der Entscheidung, ob der vorliegende Flugzeugentwurf durchgeführt oder verworfen wird. Dazu ist es notwendig während des Vorentwurfs Zuversicht für den Entwurf zu bekommen. Es muss zuverlässig eingeschätzt werden, ob der Entwurf in der geplanten Zeit und zu den geplanten Kosten entwickelt und produziert werden kann.

Die Vorentwurfsphase kann z.B. durch die Vorentwurfssoftware PrADO (Preliminary Aircraft Design and Optimization) unterstützt werden. PrADO wurde an der Technischen Universität Braunschweig entwickelt und ist in verschiedene Module aufgeteilt. Als Eingabe bekommt PrADO Daten über die Transportmission, den Grundentwurf und einzuhaltende Vorschriften. Anhand der Hauptprogrammmoduln kann der Flugzeugentwurf je nach Bedarf durchgeführt werden. Als Ergebnis liefert PrADO, je nach verwendeten Modulen, Daten über die Geometrie, das Flugverhalten, die Massenverteilung sowie über Kosten (z.B. direkte Betriebskosten - DOC). PrADO speichert die Ergebnisse der Berechnungen in disziplinaren Datenbanken, welche in einer einzigen ASCII Datenbank zusammengefasst werden.

2.2.3 Detailentwurf

Wenn die Zuversicht in den Entwurf am Ende des Vorentwurfs ausreicht, wird mit dieser Phase begonnen in der alle Komponenten bis ins letzte Detail ausgearbeitet werden. Dieser Punkt ist ein kritischer Punkt da an dieser Stelle der Großteil der Kosten entsteht, Abbildung 2.2 zeigt die festgelegten Kosten des Entwurfs gegenüber den

entstandenen Kosten. Sowohl in der Konzept-, als auch in der Vorentwurfsphase wird

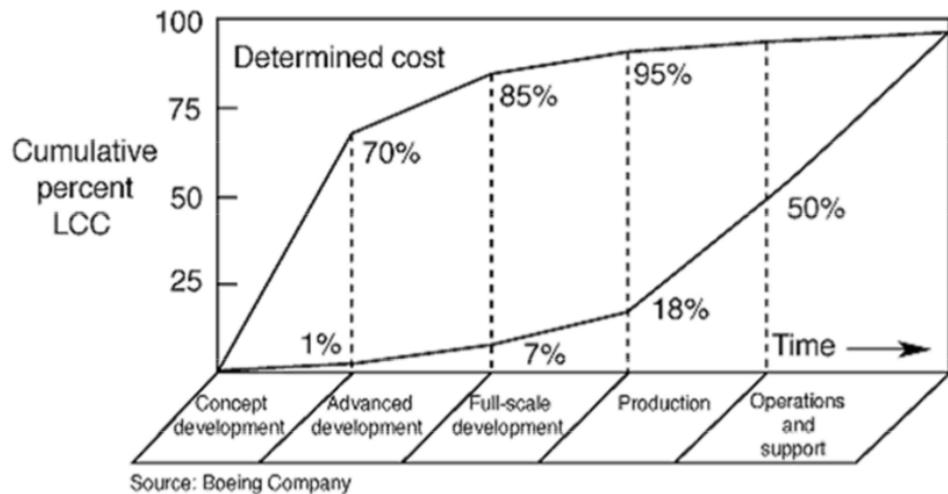


Abbildung 2.2: Kostenentwicklung im Entwurfsprozess [La 11]

durch Entscheidungen ein Großteil der Kosten festgelegt. Die Kosten entstehen allerdings erst im Detailentwurf, bzw. während der Produktion. Eine späte Entdeckung von Fehlentscheidungen/Fehleinschätzungen können große Kosten verursachen. Bisher getroffene Entscheidungen über den Flugzeugentwurf können daher, wenn überhaupt, nur noch mit erheblichem Aufwand geändert werden. Kritisch ist vor allem, wenn Anforderungen aus dem Bereich der Fertigung die erst in dieser Phase eingehen, Änderungen am Gesamtentwurf nötig machen. Fertigungsanforderungen gehen daher mit Kompromissen einher und dürfen das Gesamtkonzept nicht zu stark beeinflussen.

Subsysteme des Flugzeugentwurfs werden während der Entwicklung immer wieder separat getestet. Da theoretische Analysen mittels Software nicht ausreichen, werden einige kritische Komponenten, bereits produziert und in praktischen Versuchen untersucht. Um die Schnittstellen zwischen den verschiedenen Subsystemen zu verifizieren, werden Integrationstest durchgeführt. Dazu wird ein sogenannter „iron bird“¹ verwendet.

In dieser Phase wird sehr viel mit CAD-Anwendungen gearbeitet um die einzelnen Komponenten, Baugruppen und Bauteile auszuarbeiten. Auch CAM-Tools (Computer Aided Manufacturing) finden in dieser Phase Anwendung um die Fertigungsvorgänge zu planen. Der Detailentwurf endet mit der Auslieferung des ersten aus dem Entwurf hervorgehenden Flugzeuges.

Die in diesem Abschnitt aufgeführten Beschreibungen des traditionellen Flugzeugentwurfsprozesses sind an [Ray99] angelehnt. [Ray99] ist ein altes und sehr verbreitetes Werk über den Konzept- und Vorentwurf von Flugzeugen. [Fie99] bietet ebenfalls eine

¹ Ein „iron-bird“ ist ein Prüfstand für die Systemintegration. Er kann physikalisch, virtuell oder in Form einer erweiterten Realität (Augmented Reality) umgesetzt werden.

gute Übersicht über typische Flugzeugentwurfsprozesse. In dem aktuelleren Werk [Kun10] wird neben dem Flugzeugentwurfsprozess auch die Vorbereitung der Fertigung behandelt.

Der traditionelle Entwurfsprozess birgt zwei für moderne Flugzeugentwürfe erhebliche Nachteile. Zum Einen stützt er sich, wie Abbildung 2.3 zeigt, im Konzeptentwurf hauptsächlich auf die Aerodynamik und den Antrieb. Dadurch werden viele andere Disziplinen,

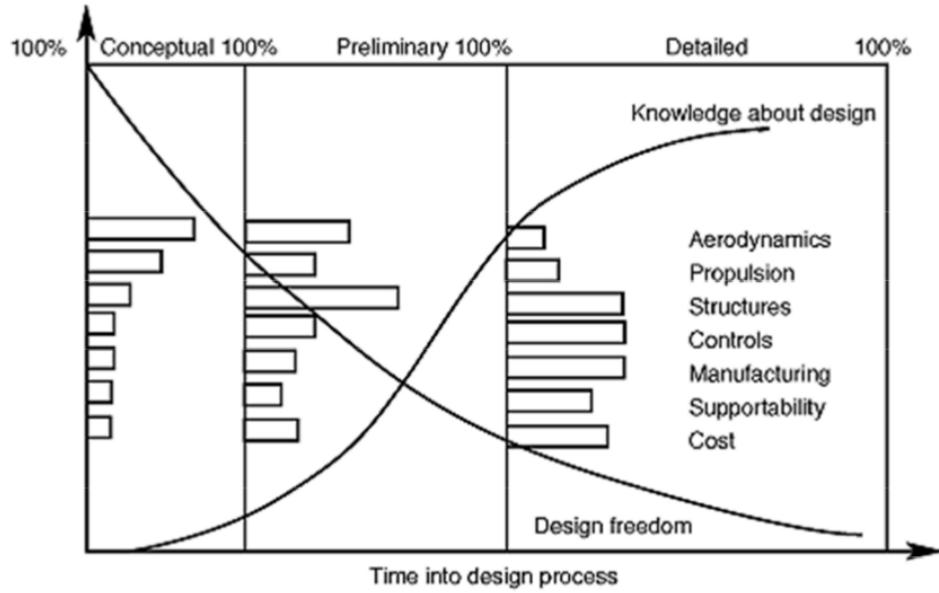


Abbildung 2.3: Verteilung der Disziplinen im traditionellen Entwurfsprozess [La 11]

deren Relevanz aufgrund ständig wachsender Anforderungen erheblich zunimmt, vernachlässigt. Dies führt über nachträgliche Änderungen zu steigenden Kosten.

Zum Anderen wird aufgrund der statistischen Entwurfsmethoden lediglich der Entwurf konventioneller Flugzeugkonfigurationen vereinfacht. Der Entwurf unkonventioneller Flugzeugentwürfe anhand dieser Methoden ist, wegen der fehlenden Datenbasis, mit großer Vorsicht zu betrachten.

Um diese Defizite auszugleichen wird an Möglichkeiten zur Umsetzung eines ganzheitlichen Entwurfsprozesses gearbeitet. Der ganzheitliche Entwurfsprozess wird im folgenden Abschnitt 2.3 beschrieben.

2.3 Ganzheitlicher Flugzeugentwurfsprozess

Wie in Abschnitt 2.2 bereits erläutert wurde fließen im traditionellen Entwurfsprozess einige Disziplinen, wie z.B. die Fertigung erst sehr spät in den Entwurf mit ein. Die erstmalige Berücksichtigung von Disziplinen in einer Phase in der der Gesamtentwurf bereits festgelegt ist und nur noch marginale Änderungen am Entwurf möglich sind, kann erhebliche Kosten verursachen und/oder die Güte des Entwurfs beeinträchtigen. Um dem entgegenzusteuern müssen alle relevanten Disziplinen bereits zu Beginn des Entwurfsprozesses berücksichtigt werden. Abbildung 2.4 zeigt, wie sich die Kurven für

den Gestaltungsfreiraum und für das Wissen über den Entwurf durch den ganzheitlichen Entwurfsprozess nach Einschätzungen verändern [La 11].

Für die Umsetzung ist es notwendig die Güte eines Entwurfs bezüglich bestimmter

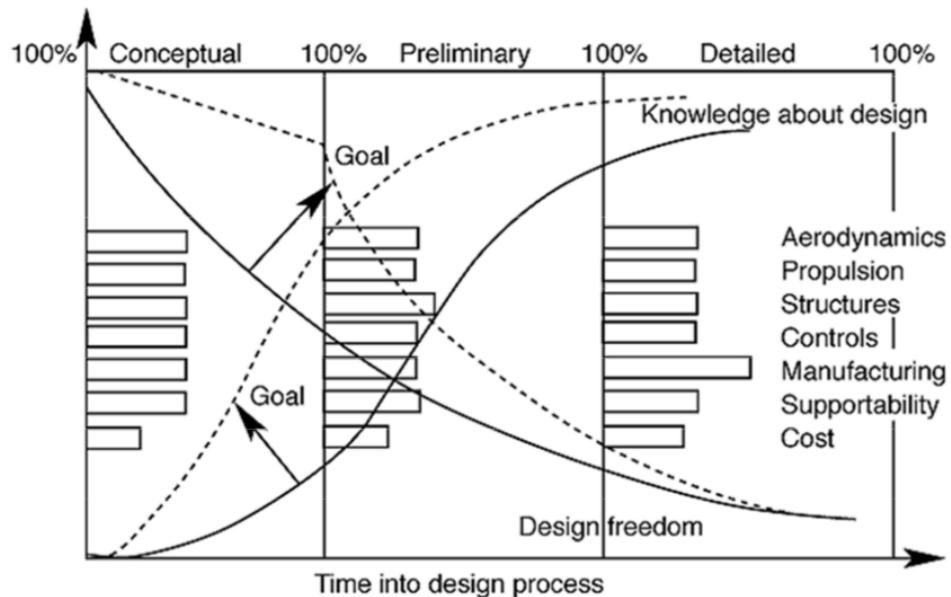


Abbildung 2.4: Verteilung der Disziplinen im ganzheitlichen Entwurfsprozess [La 11]

Disziplinen frühzeitig analysieren und bewerten zu können. Dies setzt wiederum einen ausreichend hohen Informationsgehalt des Entwurfs voraus. Für einen ganzheitlichen Entwurf ist der Informationsgehalt im Konzeptentwurf bereits mit dem des Vorentwurfs im traditionellen Flugzeugentwurfsprozess vergleichbar. Da zu Beginn jedoch eine große Menge an Konfigurationen untersucht werden, fallen erheblich mehr Daten an als im traditionellen Konzeptentwurf. Durch eine wachsende Automatisierung der Entwurfsprozesse kann die Arbeitslast dennoch auf einem angemessenen Level gehalten werden. Vor allem nicht kreative, repetitive Aufgaben, die sonst manuell von Entwicklern durchgeführt werden, können durch die Verwendung von „Knowledge Based Engineering“ automatisiert werden. „Knowledge Based Engineering“ (KBE) kann mit „Wissensbasiertes Entwickeln/Konstruieren“ übersetzt werden und wird in [La 11] wie folgt definiert (frei übersetzt):

Definition:

Knowledge Based Engineering (KBE) ist eine Technologie die auf dedizierten Softwaretools basiert, den sogenannten KBE Systemen, die in der Lage sind Produkt- und Prozess-Wissen zu sammeln und wiederzuverwenden. Das Hauptziel von KBE ist die Reduktion der Dauer und Kosten von Entwicklungszyklen anhand von:

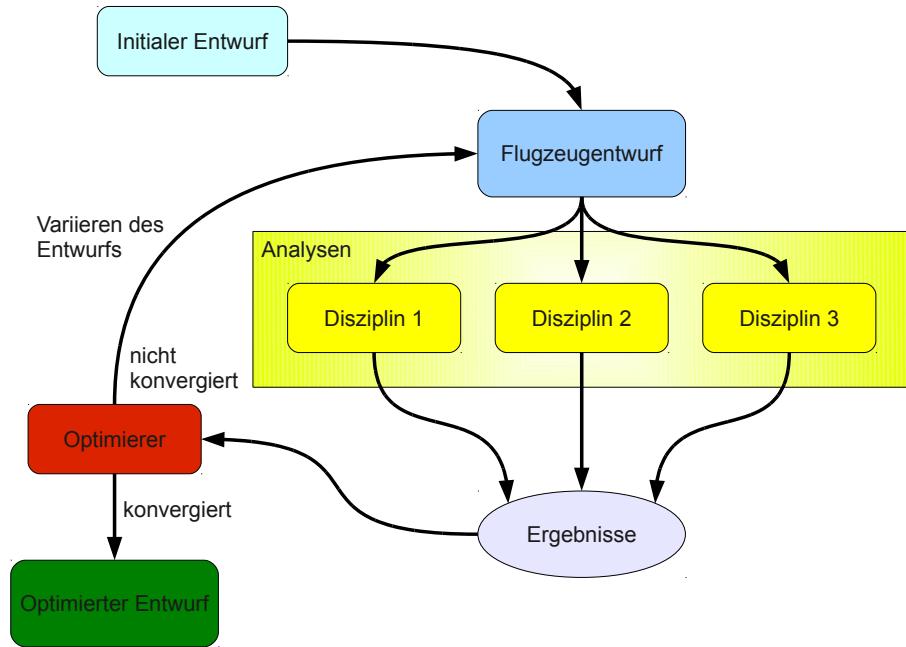
- Automatisierung von repetitiven, nicht-kreativen Entwurfsaufgaben
- Unterstützen von Multi-Disziplinärer Entwurfsoptimierung in allen Phasen des Entwurfsprozesses

In [La 11] werden Möglichkeiten für den Einsatz von KBE im Flugzeugentwurfsprozess anhand des Konzepts einer Design and Engineering Engine (DEE) vorgestellt. Dieses Konzept wird in Abschnitt 2.3.2 näher erläutert.

2.3.1 MDO

Der Kern des ganzheitlichen Entwurfsprozesses ist die Multi-Disziplinäre Entwurfsoptimierung (MDO). Die MDO ist ein Optimierungsverfahren bei dem eine Optimierung nicht nur auf eines, sondern auf mehrere Kriterien gleichzeitig durchgeführt wird. Dadurch können Optima gefunden werden, welche die besten Kompromisse zwischen den angegebenen Kriterien bieten. Mit einer MDO kann ein Flugzeugentwurf erreicht werden, der ein globales Optimum und nicht nur ein lokales (in einem oder wenigen Kriterien optimales) Optimum darstellt. Abbildung 2.5 zeigt den schematischen Ablauf einer MDO. Ausgehend von einem initialen Entwurf werden Analysen aus unterschiedlichen Disziplinen durchgeführt. Die Ergebnisse der Analysen werden an einen Optimierer übermittelt, der überprüft, ob der Entwurf den Anforderungen entsprechend konvergiert ist. Sollte dies der Fall sein, ist die Optimierung abgeschlossen und der MDO-Prozess wird beendet. Andernfalls wird anhand eines geeigneten Optimierungsverfahrens der zu Grunde liegende Entwurf verändert und die Analysen erneut durchgeführt. Die MDO ist bereits seit vielen Jahren Thema von wissenschaftlichen Arbeiten. Eine Implementierung dieses Prozesses (pyMDO) wird z.B. in [Alo04] beschrieben. pyMDO ist ein Softwareframework für eine MDO detaillierter Flugzeugentwürfe. Das Framework ist in Python implementiert und ermöglicht die Anbindung verschiedener Analyse- und Optimierungstools. Durch Verwendung von objektorientierter Programmierung erreicht pyMDO einen hohen Grad an Modularität, was die Anbindung zusätzlicher Tools vereinfacht. Es werden „abstrakte“ Klassen verwendet um Schnittstellen für unterschiedliche Komponenten zu definieren, wie z.B. Optimierer, Analysierer etc.. Über Vererbung werden die Schnittstellen zu den konkreten Tools implementiert [Alo04].

pyMDO bietet somit die Möglichkeit Tools aus verschiedenen Disziplinen (Aerodynamik, Struktur, Geometrie, Visualisierung) miteinander zu verbinden und somit einen Flugzeugentwurfsprozess zu ermöglichen, der je nach Bedarf verschiedene Disziplinen

**Abbildung 2.5:** MDO-Prozess

berücksichtigt. Es ermöglicht unter Anderem die Verwendung der in Tabelle 2.1 angegebenen Tools [Alo04]. Außerdem beinhaltet pyMDO ein Modul zum Erstellen von

SNOPT	SNOPT ist ein gradienten-basierter Optimierer der großformatige, nicht-linear-abhängige Optimierungsprobleme löst.
TFLO2000	TFLO2000 ist eine Software zur Lösung von Strömungsproblemen, welcher die Verwendung von Finiten-Elementen, einem modifizierten Runge-Kutta Verfahren und einigen anderen Verfahren ermöglicht.
FEAP	FEAP (Finite Element Analysis Program) ist ein Programm für die Strukturanalyse auf Basis von Finiten-Elementen.

Tabelle 2.1: In pyMDO integrierte Tools

unterschiedlichen Meshes für weitere Analysetools (Aerosurf).

2.3.2 Entwurfsumgebungen

Wesentlich für die Durchführung einer MDO ist der Zusammenschluss von verschiedenen Softwareprogrammen zu einem Entwurfsprozess. Zu diesem Zweck können Entwurfsumgebung mit integrierter Workflow Engine verwendet werden. Da in einem modernen Flugzeugentwurf Kenntnisse über viele unterschiedliche Disziplinen benötigt werden, ist es sinnvoll Experten in den Entwurfsprozess zu integrieren.

Das DLR besteht aus 32 Instituten die an 16 Standorten in ganz Deutschland verteilt

sind¹. Jedes Institut hat seinen eigenen Schwerpunkt und somit Wissen, welches es zum Entwurfsprozess beisteuern kann. Um die Institute und somit das Wissen zu vernetzen entwickelt das DLR, in Kooperation mit dem Fraunhofer-Institut für Algorithmen und Wissenschaftliches Rechnen SCAI, eine komponentenbasierte verteilte Plattform zur Integration von Applicationen². Die RCE (Remote Component Environment) Software, die eine Workflow Engine enthält, ermöglicht es Entwurfsprozesse als verteiltes System zu realisieren. In RCE können über eine API (Application Programming Interface) Programme eingebunden und anderen Benutzern über das Netzwerk zur Verfügung gestellt werden. Ein wesentlicher Vorteil eines solchen Systems ist, dass die Programme der unterschiedlichen Disziplinen bei den Experten auf den jeweiligen Gebieten bleiben. Außerdem wird die Zusammenarbeit zwischen den Instituten stark vereinfacht. Die Institute können über RCE gemeinsam an einem Projekt arbeiten und ihr Wissen noch einfacher in das Projekt einbringen. Auch die Daten, die in einem Entwurfsprozess auftreten können über das verteilte System verwaltet werden. Für CPACS wurde Chameleon/RCE entwickelt, welches ein um CPACS-Funktionalitäten erweitertes RCE darstellt. In Chameleon/RCE sind Module implementiert, die den Zugriff auf und die Verwendung von CPACS-Daten im Entwurfsprozess vereinfachen. Chameleon/RCE ist sowohl unter Windows, als auch auf Unixsystemen lauffähig und bietet zur Interaktion ein grafisches Benutzerinterface (siehe Abbildung 2.6).

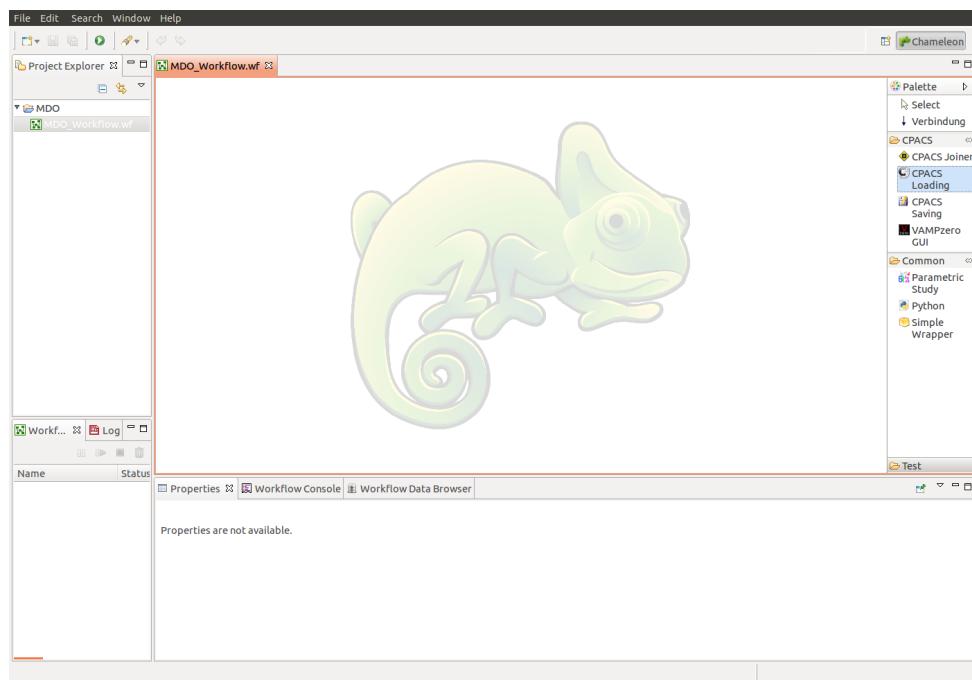


Abbildung 2.6: Chameleon/RCE Benutzerinterface

1 www.dlr.de/dlr/desktopdefault.aspx/tabcid-10443/637_read-251/

2 www.dlr.de/sc/desktopdefault.aspx/tabcid-5625/9170_read-17513/

Eine weitere Herausforderung auf dem Weg zur Automatisierung ist es, das Wissen von Flugzeugentwerfern zu digitalisieren und in Programmen verfügbar zu machen. Daher besteht eine Notwendigkeit zum Erstellen wissensbasierter Methoden zum Generieren von Daten auf Grundlage von Entscheidungssystemen (Knowledge Based Engineering). Ein modernes Konzept, das ebenfalls eine MDO ermöglicht und sich mit dem Thema wissensbasierter Methoden für den Flugzeugentwurf beschäftigt, ist die an der TU Delft entwickelte Design and Engineering Engine (DEE) [La 11]. Das Konzept einer DEE beschreibt ein System für die multidisziplinäre Erstellung, Analyse und Optimierung von Flugzeugentwürfen. Abbildung 2.7 zeigt die allgemeine Struktur einer DEE. Ein Flugzeug-

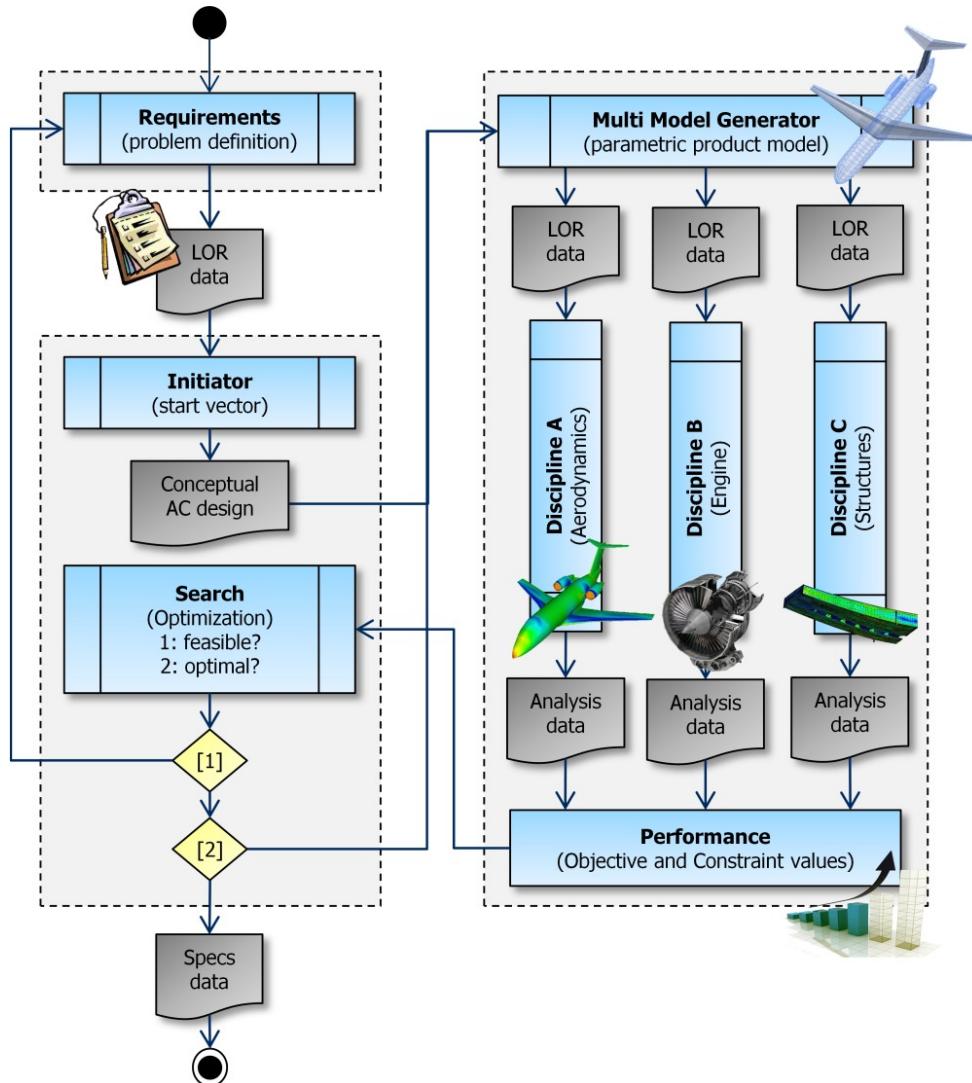


Abbildung 2.7: Struktur einer Design and Engineering Engine [La 11]

entwurf beginnt in der Regel mit einer Liste von Anforderungen, welche zu Beginn des Entwurfsprozesses definiert werden. Diese werden dann von einem Initiierer verwendet um einen ersten konzeptionellen Flugzeugentwurf zu erstellen. Die darauf folgenden

Elemente bilden das Kernstück der DEE. Ein sogenannter Multi-Model-Generator (MMG) erstellt aus dem initialen Entwurf ein parametrisches Produktmodell. Dieses Produktmodell stellt die Basis des anschließenden Entwurfsprozesses dar. Der MMG enthält Module, die aus dem parametrischen Modell z.B. geeignete FEM- oder CFD-Modelle generieren (daher der Name „Multi-Model-Generator“). Diese generierten Modelle können dann von den entsprechenden Analysetools verwendet werden um die Güte des Flugzeugentwurfs zu beurteilen. Die Ergebnisse der Analysetools werden dann mit Entwurfszielen verglichen und festgestellt, ob der Entwurf erstens realisierbar und zweitens optimal ist. Falls der Entwurf nicht realisierbar ist, müssen die Anforderungen überarbeitet werden und der ganze Prozess erneut ausgeführt werden. Wenn der Entwurf realisierbar, aber noch nicht optimal ist, wird das parametrische Modell durch den Optimierer angepasst und erneut analysiert. Bei Änderungen des parametrischen Modells, welche die Modelle folgender Entwurfsprogramme betreffen, werden diese Modelle vom MMG erneut generiert. Auf diese Weise ist sichergestellt, dass alle Analyseprogramme immer auf aktuelle Daten zugreifen. Da alle Datenmodelle auf dem parametrischen Modell aufbauen ist die Konsistenz der Daten gewährleistet.

An der TU Delft wird das Konzept der DEE umgesetzt und an einer konkreten Implementierung einer DEE gearbeitet. Der Schwerpunkt der Entwicklung liegt dabei in der Implementierung des MMG. Die Entwicklung des MMG wurde in den letzten Jahren im Zuge mehrerer Masterarbeiten durchgeführt [vH09], [vdB09], [Kon10], [Bro11]. Der MMG umfasst zur Zeit Parametrisierungen für Tragflächen und Rumpfkomponenten und wird ständig weiterentwickelt. Auch innerer Strukturen und Klappensysteme können modelliert werden. Der MMG ermöglicht die Verbindung zu Analysesoftware wie ESDUpac 9931, MSES 3.04 [vdB09] und VS Aero [Kon10] zur aerodynamischen Analyse, und zu NASTRAN für die Strukturanalyse [vH09]. Außerdem kann die äußere Flugzeuggeometrie wahlweise nach IGES oder STEP exportiert werden [vdB09]. Die Implementierung des MMG wird in GDL¹ durchgeführt.

Das Konzept einer DEE bietet eine gute Möglichkeit die Konsistenz der Entwurfsdaten zu gewährleisten indem die Analysetools in jedem Iterationsschritt aktualisierte Daten vom MMG bekommen. Der Anschluss unterschiedlicher Analysetools an den MMG ist allerdings relative aufwändig. Im Folgenden wird eine Möglichkeit beschrieben den Aufwand durch Verwendung einer zentralen Schnittstelle zu verringern.

2.3.3 Zentrale Schnittstelle

Um die verschiedenen Entwurfs- und Analysetools von Spezialisten in einer Entwurfsumgebung zu verwenden und miteinander verbinden zu können, ist es notwendig den Datenaustausch zwischen den einzelnen Tools zu ermöglichen. In einem System, dass

¹ GDL ist eine auf LISP basierende Programmiersprache. Sie beinhaltet CAD-Funktionalitäten und ermöglicht die Entwicklung von wissensbasierten Systemen.

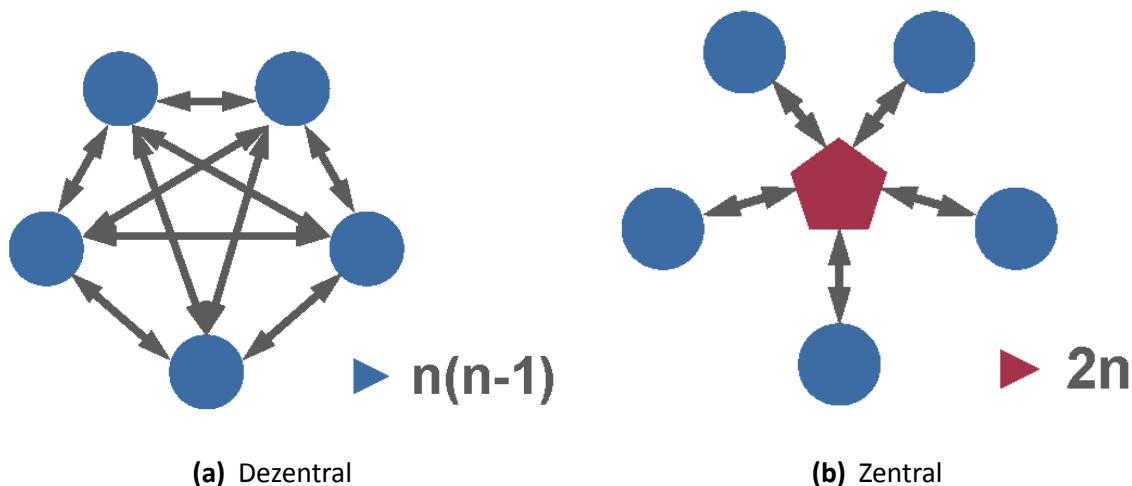


Abbildung 2.8: Vergleich dezentrale und zentrale Schnittstelle

aus n Tools besteht und bei dem jedes Tool mit jedem verbunden ist, sind $n \cdot (n - 1)$ Schnittstellen notwendig, da die Verbindung zweier Tools jeweils zwei Schnittstellen (Umwandeln von Daten A nach B und umgekehrt) beinhaltet. Abbildung 2.8a veranschaulicht dies für ein System bestehend aus fünf Tools. Durch die Verwendung einer zentralen Schnittstelle sinkt die Anzahl der benötigten Schnittstellen erheblich, da sich jedes Tool lediglich mit der zentralen Schnittstelle verbinden muss. Das reduziert die notwendigen Schnittstellen, wie in Abbildung 2.8b dargestellt auf $2 \cdot n$. Der nunmehr lineare Anstieg von Schnittstellen durch Hinzufügen von Tools zu einem System mit zentraler Schnittstelle erhöht deren Skalierbarkeit erheblich.

Wesentlich für eine zentrale Schnittstellendefinition ist es, alle zu einem Themengebiet gehörenden Daten zu enthalten und Redundanzen zu vermeiden. Zur Definition einer zentralen Schnittstelle für die Luftfahrt, können unterschiedlichen Informationsmodelle verwendet werden.

Hinweis:

Die Vorteile einer zentralen Schnittstelle sind keineswegs eine neue Erkenntnis. Bereits seit ca. einem Jahrhundert existiert das Deutsche Institut für Normung, welches es sich zur Aufgabe gemacht hat sinnvolle Normen festzulegen, um die Zusammenarbeit zwischen Menschen/Organisationen/Einrichtungen zu vereinheitlichen. Standards bringen Struktur in eine immer komplexer werdende Umgebung. Sie ermöglichen es modulare Strukturen aufzubauen in denen eine Austauschbarkeit von einzelnen Modulen gewährleistet ist.

In [Böh09] wird die Eignung dreier unterschiedlicher Informationsmodelle für die Datenintegration in den Entwurfsprozess untersucht. Die dort betrachteten Informationsmodelle sind:

- das am DLR entwickelte CPACS
- das von der ISO definierte STEP (ISO 10303)
- die aus der Softwareentwicklung stammende UML

Das „Common Parametric Aircraft Configuration Schema“ (CPACS) ist ein seit 2005 am DLR entwickeltes Datenformat. Es ist eine XML basierte, hierarchisch angeordnete Struktur mit Daten zum gesamten Lebenszyklus von Luftfahrzeugen. Da CPACS wesentlicher Bestandteil dieser Arbeit ist, wird es in Abschnitt 3.2 detaillierter erläutert.

STEP (Standard for the Exchange of Product model data) ist ein in ISO 10303 definierter Standard zur Beschreibung von Produktdaten. STEP wird bereits seit 1984 von der Internationalen Organisation für Standardisierung (ISO) entwickelt. Da STEP mit der Anforderung entwickelt wird alle möglichen Produkte mit sämtlichen Informationen abzubilden, sind die Spezifikationen sehr umfangreich und generisch. In STEP können ebenfalls Daten zum gesamten Lebenszyklus von Produkten abgebildet werden. STEP bietet inzwischen auch die Möglichkeit Daten im XML-Dateiformat zu verwenden.

Die „Unified Modeling Language“ (UML) ist eine objektorientierte Modellierungssprache. Ihre Entwicklung begann bereits 1990 und im Jahr 2005 wurde von der OMG (Object-Management-Group) UML2 veröffentlicht. UML ermöglicht es Zusammenhänge von Informationen in Strukturdigrammen zu beschreiben.

Der Author von [Böh09] kommt zu dem Ergebnis, dass STEP aufgrund der Größe/Komplexität und des hohen Abstraktionsgrades für die Verwendung zur Beschreibung eines zentralen Datenmodells im DLR ungeeignet ist. Sowohl UML, als auch das auf XML basierte CPACS bieten geeignete Möglichkeiten zur Beschreibung von Daten für den Flugzeugentwurf. UML bietet, von den untersuchten Informationsmodellen, die fortschrittlichsten Modellierungsmöglichkeiten für Flugzeugentwurfsdaten. XML wiederum sticht durch eine weite Verbreitung und einfache Verwendbarkeit hervor. Außerdem ermöglicht XML einen einfachen Datenaustausch mit anderen Formaten [Böh09].

Die Arbeit von Böhnke [Böh09] unterstreicht den Wert der Entwicklung von CPACS am DLR. Inzwischen ist CPACS innerhalb des DLRs bereits stark verbreitet und wird in verschiedenen Projekten verwendet. Einige dieser Projekte sind im Folgenden aufgeführt¹.

VAMP VAMP steht für „Virtual Aircraft Multidisciplinary Analysis and Design Processes“ und beschäftigt sich mit der Entwicklung eines numerischen Gesamtentwurfssystems für Flugzeuge auf Vorentwurfsniveau. Es ist das Nachfolgeprojekt von TIVA I/II.

TIVA I/II TIVA steht für „Technology Integration for the Virtual Aircraft“ und behandelte die Integration von Entwurfswerkzeugen in eine gemeinsame Softwareumgebung. Dabei wurde CPACS als einheitliche Schnittstelle zwischen den Tools verwendet.

¹ Die Projekte werden im VAMP Projektplan kurz vorgestellt und sind diesem entnommen [Deu09].

UCAV UCAV steht für „Unmanned Combat Air Vehicle“ und beschäftigt sich mit der Entwicklung unbemannter Systeme die zunehmend autonom agieren können.

EVITA EVITA steht für „Evaluierung innovativer Turboantriebe“ und ist ein Projekt, das sich mit der Entwicklung eines Entwurfssystems für Turboantriebe beschäftigt. Ziel dieses Systems ist unter anderem die Reduktion Treibstoffverbrauch und Lärmemissionen wie sie in ACARE 2020 angestrebt werden.

CATS CATS steht für „Climate Compatible Air Transportation System“ und beschäftigt sich mit den Auswirkungen des Luftverkehrs auf das Klima.

Auch außerhalb des DLR findet CPACS bereits Verwendung. Z.B. wird an der TU Delft zur Zeit an einer Verbindung des MMG mit der CPACS-Schnittstelle gearbeitet, um die Vorteile von CPACS und den um CPACS entwickelten Softwaretools in der DEE zu nutzen. Diese Schnittstelle ermöglicht es Analysetools mit einer CPACS-Schnittstelle direkt in der DEE zu verwenden. Der weiter unten beschriebene „The Initiator“ ermöglicht ebenfalls einen Export der Geometrien und einiger Performance-Daten nach CPACS.

Auch an der Königlichen Technischen Hochschule in Stockholm (KTH) wird CPACS bereits verwendet. In [Riz12] wird vorgestellt, wie unter anderem an der KTH entwickelte Softwaretools über die CPACS-Schnittstelle in einer Entwurfsumgebung eingesetzt werden können.

2.3.4 Analysetools

Die Anzahl an Analysetools für den Flugzeugentwurf ist enorm. Für diese Arbeit ist lediglich der Zweck von Analysetools entscheidend und, dass sie über geeignete Modelle (teilweise toolspezifisch) verwendet werden können. An dieser Stelle wird daher nur ein Analysetool (Tornado) vorgestellt, welches zum Testen des im Zuge dieser Arbeit entwickelten Simple Geometrie Generators in Kapitel 4 verwendet wird.

Tornado ist ein Softwaretool zur aerodynamischen Analyse von Tragflächenkonfigurationen. Es ist eine Matlab-Implementierung der „Vortex Lattice Methode“ und wurde von Tomas Melin im Zuge seiner Masterarbeit an der KTH implementiert [Mel00]. Tornado steht unter der GNU-General Public License, wodurch es möglich ist es an individuelle Bedürfnisse anzupassen und/oder Zusatzmodule zu implementieren. Die Eingabe erfolgt über eine Input-Datei oder interaktiv über die Matlab-Konsole. Die an Tornado übermittelten Daten werden von einem Preprozessor verarbeitet, der die Geometrie in ein geeignetes Gitternetz überführt. Seit der ersten Veröffentlichung von Tornado im Jahr 2001 wurde es regelmäßig weiterentwickelt und z.B. auch um die Berechnung von instationären Strömungen erweitert [Mel06]. Tornado ermöglicht eine Analyse von im dreidimensionalen Raum frei orientierten Auftriebsflächen und akzeptiert die verschiedensten Tragflächenformen. Es berechnet für die eingegebenen Geometrien und Umgebungsbedingungen (wie z.B. Machzahl und Anstellwinkel) die aerodynamischen Koeffizienten (Auftriebsbeiwert, Widerstandsbeiwert, Nullwiderstandsbeiwert und Momentenbeiwert).

Durch einen am DLR entwickelten CPACS-Wrapper kann Tornado verwendet werden um CPACS-Geometrien zu Analysieren. Der Wrapper übersetzt die CPACS-Geometrien in das von Tornado benötigte EingabefORMAT.

2.3.5 Initialisierung

Wie bei allen Optimierungsproblemen muss auch für eine MDO ein Startpunkt angegeben werden. Im Flugzeugentwurfsprozess muss daher zunächst ein Flugzeugentwurf initiiert werden. Ein initialer Flugzeugentwurf für eine MDO kann auf unterschiedliche Weise erstellt werden. Eine Möglichkeit ist es ein Konzeptentwurfstool zu verwenden und den damit erstellten Entwurfs als Startpunkt für die MDO zu verwenden. Für eine Initialisierung können z.B. die Konzeptentwurfstools VAMPzero und „The Initiator“¹, oder das in Abschnitt 2.2.2 vorgestellte Vorentwurfstool PrADO verwendet werden. Über das Konvertertool reShapeP2C kann die von PrADO erstellte äußere Flugzeuggeometrie in das CPACS-Format konvertiert werden [Jep10].

Wegen der Beschränkung dieser Tools können auf diese Weise in der Regel nur konventionelle Flugzeugentwürfe generiert werden. Um auch unkonventionelle Entwürfe erstellen zu können, kann ein geometriebasierter Entwurfsansatz verwendet werden. Ein Softwaretool für den geometriebasierten Flugzeugentwurf ist z.B. OpenVSP.

Im Folgenden werden die drei Tools (PrADO wurde bereits in 2.2.2 beschrieben) zum Erstellen initialer Flugzeugentwürfe kurz vorgestellt.

VAMPzero ist ein in Python geschriebenes Konzeptentwurfstool. Es wurde bereits am 06.10.2011 auf der PyCon² in Leipzig vorgestellt und ist seit Dezember 2011 als Open Source Projekt auf vampzero.googlecode.com verfügbar.

VAMPzero ermöglicht neben dem Entwurf der äußeren Geometrie auch den Entwurf innerer Strukturen, die Dimensionierung von Triebwerken sowie eine Missions- und Kostenanalyse. Es basiert auf Handbuchverfahren, verwendet aber je nach Vorgaben unterschiedliche Berechnungsmethoden, was es sehr flexibel macht. Um die Transparenz des durchgeföhrten Entwurfs zu erhöhen dokumentiert VAMPzero seine durchgeföhrten Berechnungen in Form von Diagrammen. Die Diagramme zeigen die Abhängigkeiten der Parameter untereinander und somit welche Werte zum Berechnen bestimmter Parameter verwendet wurden. Zum Konfigurieren des Flugzeugentwurfs kann ein grafisches Benutzerinterface verwendet werden. VAMPzero bietet aber auch einfache Erweiterungsmöglichkeiten durch das Schreiben von eigenen Skripten in Python. Die Eingabedaten werden aus einer CPACS-Datei gelesen und die Ergebnisse ebenfalls in eine CPACS-Datei geschrieben.

1 Da dem Programm in [Lan11] kein Name gegeben wird, wird es in dieser Arbeit, auf Vorschlag des Authors, als „The Initiator“ bezeichnet.

2 Python Konferenz

Aufgrund der Möglichkeit VAMPzero im Batch-Mode auszuführen eignet es sich auch für Parameterstudien. Diese könnten zum Beispiel mit Chameleon durchgeführt werden, in welches VAMPzero einfach integriert werden kann. Auch das grafische Benutzerinterface ist ebenfalls in Chameleon integriert und ermöglicht eine Verwendung ohne separate CPACS-Datei¹.

„The Initiator“ ist ein in Matlab implementiertes Konzeptentwurfstool. Es ist aus den folgenden drei Modulen aufgebaut:

1. Initiierer
2. Analysierer
3. Optimierer

Der Initiierer ermittelt Anfangswerte für einen gewählten Entwurf, die auf Basis einer Entwurfsdatenbank ausgewählt werden. Die im Tool enthaltene Datenbank umfasst Daten von 55 Flugzeugentwürfen. Der Analysierer ermöglicht die Untersuchung des Flugzeugentwurfs. „The Initiator“ enthält Möglichkeiten zur Massen- und Schwerpunktsabschätzungen, zur aerodynamische Analyse, zur Prüfung von Stabilität und Steuerbarkeit sowie für Reichweitenuntersuchungen. Der Optimierer, bestehend aus einem maßgeschneiderten genetischen Algorithmus für mehrere Entwurfsziele und Bedingungen, optimiert den Entwurf auf unterschiedliche Entwurfsziele. Der Optimierer ist besonders für den Entwurf von Box-Wing-Konfigurationen geeignet. Verwendet wird „The Initiator“ durch ein grafisches Benutzerinterface, welches den Benutzer durch die Entwurfsphasen führt. Analyseergebnisse werden dem Benutzer sowohl als Werte, aber auch in Form von Graphen/Diagrammen präsentiert. „The Initiator“ ist auf den Entwurf von traditionellen und Box-Wing-Konfigurationen beschränkt [Lan11].

Der an der Technischen Universität in Delft (TUDelft) entwickelte „The Initiator“ wurde inzwischen mit einer CPACS-Schnittstelle ausgerüstet und ermöglicht es Entwürfe nach CPACS zu exportieren.

OpenVSP ist ein parametrisches Flugzeugentwurfstool. Es ist die Open Source Variante des ursprünglichen VSP (Vehicle Sketch Pad), welches bereits seit 1990 von J.R. Gloudemans u.A. für die NASA entwickelt wird. OpenVSP wurde am 10 Januar 2012 auf www.openvsp.org veröffentlicht.

Durch die Verwendung von parametrischer Geometrie wird der Modellierungsprozess stark beschleunigt. Außerdem verbessert es die Möglichkeit für Entwickler frühzeitig mit high-fidelity Analysen zu arbeiten indem es Exportmöglichkeiten für andere Softwaretools, wie z.B. CFD zur Verfügung stellt. Die Möglichkeiten zur Erzeugung von

¹ Die benötigte CPACS-Datei wird von der GUI erstellt.

CFD-Daten mit OpenVSP wurde seit der ersten Version weiter verbessert [Hah12]. OpenVSP ermöglicht auch den Import von Geometrien aus anderen Datenformaten. Eine komplette Liste der Import-/Exportmöglichkeiten befindet sich im Internet unter www.openvsp.org/wiki/doku.php?id=representations.

Wenn OpenVSP als zentrales Entwurfswerkzeug im Konzeptentwurf verwendet wird, können Modelle für viele unterschiedliche Analysetools generiert werden. Durch die Automatisierung dieser sonst zeitraubenden Tätigkeiten, können viele high-fidelity Analysen bereits in der Konzeptphase verwendet werden um die Güte eines Entwurfs zu ermitteln. Zwischen zwei Iterationsschritten muss lediglich die Geometrie in OpenVSP angepasst werden. Modelle für Analysen können dann wieder automatisiert erstellt werden. Auf diese Weise ist sichergestellt, dass den Analysetools immer die aktuelle Flugzeuggeometrie zur Verfügung steht. Der geringe Zeitaufwand zur Anpassung von Entwürfen zwischen zwei Iterationen ist ein Schlüsselmerkmal von OpenVSP [Fre10].

OpenVSP enthält elf vordefinierte parametrische Komponenten anhand derer Flugzeugentwürfe erstellt werden können. Dazu gehören Triebwerke, Triebwerksgondeln, Flügelkomponenten (Tragfläche, Leitwerke), Rümpfe und einige mehr. Die Parameter sind typische Entwurfsparameter, sodass eine weitgehend intuitive Verwendung von OpenVSP für Benutzer mit Kenntnissen im Bereich des Flugzeugentwurfs möglich ist. Neben den äußereren Geometrien können mit OpenVSP auch Rippen und Holme parametrisch positioniert werden [Cha12].

Zusätzlich zu den parametrischen Komponenten beinhaltet OpenVSP Bibliotheken mit Sammlungen von Profilen, Komponenten und ganzen Flugzeugentwürfen. Diese können geladen oder in einen bestehenden Entwurf eingefügt werden.

Um die Modellierung von Entwürfen auf Basis von Draufsicht, Vorderansicht und Seitenansicht zu erlauben, können diese als Hintergrundbilder in die Ansicht von OpenVSP geladen werden.

2.4 Zusammenfassung/Fazit

In diesem Kapitel wurde gezeigt, wie die Entwicklung der Luftfahrt den Flugzeugentwurfsprozess beeinflusst. Es wurden die Defizite des traditionellen Entwurfsprozesses aufgezeigt und mit dem ganzheitlichen Entwurfsprozess ein verbreiteter Lösungsansatz vorgestellt. Der ganzheitliche Entwurfsprozess, der auf einer MDO basiert, kann auf unterschiedliche Weise umgesetzt werden. Wesentlich ist die zunehmende Automatisierung, welche die Entwickler bei repetitiven, nicht-kreativen Aufgaben entlastet. Das Zusammenbringen und Organisieren von Wissen aus unterschiedlichen Disziplinen ist für den modernen Entwurfsprozess von zentraler Bedeutung. Um dies zu realisieren können wissensbasierte Entwurfsmechanismen entwickelt werden, wie sie für die Entwicklung der DEE an der TU Delft verwendet werden [La 11]. Die Mechanismen in Form von Softwaretools können in den verschiedenen Bereichen separat implementiert und gewartet werden. Um Experten unterschiedlicher Disziplinen und deren Tools zusammenzubringen,

werden verteilte Anwendungen entwickelt, die eine einfache Zusammenarbeit verschiedener Einrichtungen ermöglicht.

Am DLR wird dazu die Software RCE entwickelt, die einen MDO-Prozess ermöglicht, der aus verteilten Anwendungen zusammengesetzt und einrichtungsübergreifend überwacht werden kann. Der Datenaustausch zwischen den Anwendungen wird über die zentrale CPACS-Schnittstelle durchgeführt. Der Vorteil einer zentralen Schnittstelle gegenüber einer dezentralen Schnittstelle wurde erläutert und ist in Abbildung 2.8 dargestellt. Der Einsatz von CPACS in RCE reduziert den Aufwand der notwendig ist um verschiedene Softwaretools miteinander zu verbinden.

Auf dem Weg zu einer CPACS-basierten Entwurfsumgebung werden zur Zeit einige Softwaretools entwickelt, die mit CPACS-Datensätzen arbeiten können. Um einen Entwurfsprozess in einer CPACS basierten Umgebung zu starten stehen zur Zeit VAMPzero, „The Initiator“ und ein Konvertierungstool von PrADO nach CPACS zur Verfügung. Alle drei Tools sind geeignet um erste traditionelle Flugzeugentwürfe zu erstellen, die Anhand von Entwurfsparametern wie z.B. Reichweite und Nutzlast definiert werden. Ein wesentliches Ziel einer Entwurfsumgebung für einen ganzheitlichen Entwurfsprozess ist es auch den Entwurf unkonventioneller Flugzeugkonfigurationen zu ermöglichen. Da die genannten Tools Großteils auf statistischen Verfahren basiert, sind die Möglichkeiten für gänzlich neue Entwürfe jedoch eingeschränkt. Tools wie OpenVSP und der MMG können die Erstellten Geometrien bisher nur anhand ausgewählter Analysemodule untersuchen. Das Generieren von Flugzeugentwürfen für die CPACS-Schnittstelle ermöglicht den Anschluss an eine Reihe von Analysemodulen¹. Für CPACS existiert zur Zeit kein geeignetes Tool um, wie mittels MMG oder OpenVSP, geometriebasierte Entwürfe zu erstellen. Um einen solchen Entwurf in CPACS zu erstellen, muss die XML-Struktur von Hand mit einem XML- oder Texteditor erstellt werden. Ein Tool, welches nativ die Erzeugung und Bearbeitung von CPACS-Geometrien ermöglicht wäre daher ein wertvoller Schritt in Richtung einer CPACS basierten DEE. Ziel dieser Diplomarbeit ist daher die Entwicklung eines Tools zur interaktiven Generierung von geometriebasierten Flugzeugentwürfen (konventionell und unkonventionell) in CPACS. Die Entwicklung des SGGs (Simple Geometry Generator) wird in Kapitel 3 beschrieben.

¹ An alle Analysemodule die über eine CPACS-Schnittstelle verfügen.

3 Simple Geometry Generator (SGG)

In der Einleitung wurde erläutert, wie der Wandel des Entwurfsprozesses durch Automatisierung vorangetrieben wird und welche Rolle Softwaretools dabei spielen. Der Überblick über die aktuellen Softwareentwicklungen rund um den Flugzeugentwurf in Kapitel 2 zeigt wie Entwurfstools in einem System zusammen interagieren können um komplexe Entwurfsaufgaben zu lösen. Die Aufgabe einer zentralen Datenverwaltung wird in den bereits implementierten Frameworks und Entwurfsumgebungen deutlich. Die Entwicklung von CPACS ist ein großer Schritt in Richtung einer standardisierten Schnittstelle, die eine zentrale Datenverwaltung erheblich vereinfacht. Durch die Verwendung einer zentralen Schnittstelle werden Änderungen nur an einer Stelle durchgeführt, was zu einer großen Entlastung in der Datenverwaltung führt.

Die entstandenen Softwaretools im TIVA- und VAMP-Projekt bieten bereits vielfältige Möglichkeiten für den Umgang mit CPACS-Dateien. Vor allem steht mit VAMPzero eine Software für den Konzeptentwurf zur Verfügung, die zum Lösen typischer Entwurfsaufgaben sehr gut geeignet ist. Mit VAMPzero können so anhand weniger Entwurfsparameter, wie z.B. Reichweite, Passagierzahl, Nutzlast, etc., erste Flugzeugentwürfe erstellt und nach CPACS exportiert werden. In VAMPzero werden Großteils statistische Verfahren verwendet, die bei stark von traditionellen Konfigurationen abweichenden Entwürfen nicht, oder nur bedingt verwendet werden können. Für innovative, unkonventionelle Entwürfe müssen die ersten CPACS-Dateien von Hand¹ erstellt und bearbeitet werden. Diese Art der Geometrieerstellung ist relativ aufwendig, zeitintensiv und fehleranfällig. Daher wurde in dieser Arbeit ein Tool entwickelt mit Hilfe dessen die Erstellung geometriebasierter Entwürfe deutlich vereinfacht wird. Bei der Verwendung des SGG können zur Erstellung von Flugzeuggeometrien, typische Entwurfsparameter verwendet werden. Als Feedback dienen zum Einen drei vereinfachte zweidimensionale Ansichten des Entwurfs, und zum Anderen kann der Entwurf parallel zum SGG mittels TIGLViewer in einer dreidimensionalen Umgebung interaktiv betrachtet werden.

3.1 Anforderungen an die Software

Vor der Implementierung des SGG ist es wichtig konkrete Ziele des SGG zu definieren. Dazu wird zunächst eine Anforderungsliste erstellt, in der die Anforderungen an den SGG festgehalten werden. Im Zuge dieser Diplomarbeit wurde der SGG so implementiert, dass

¹ z.B. mit einem Plain Texteditor oder einem XML-Editor

die in diesem Abschnitt aufgeführten Anforderungen erfüllt werden. Eine tabellarische Liste mit den Anforderungen befindet sich in Anhang A.6.

Die wesentlichen Anforderungen werden im Folgenden erläutert.

Laden&Speichern Der Simple Geometry Generator (SGG) kann valide CPACS Dateien laden und speichern. Die geladenen Dateien können editiert werden. Der SGG verwendet die CPACS-Version 2.0. Die einwandfreie Funktionalität ist nur mit CPACS-Dateien gewährleistet, die zu dieser Version konform sind.

Modellauswahl Da die CPACS-Struktur die Definition mehrerer Flugzeuge erlaubt, ist für den Benutzer eine Auswahlmöglichkeit zwischen bestehenden Flugzeugentwürfen gegeben. Zusätzlich kann auch ein neuer Flugzeugentwurf, in einem neuen Modell, begonnen werden.

Neuer Entwurf Um einen komplett neuen Entwurf zu beginnen, wird ein leeres Flugzeugmodell erstellt. Zu diesem Entwurf können nach Belieben neue Komponenten (Tragflächen, Leitwerke, Rümpfe) hinzugefügt werden. Da die Leitwerke in CPACS als „wing“-Elemente realisiert sind, wird im Folgenden lediglich zwischen Rumpf- und Flügelkomponenten unterschieden.

Erstellen von Komponenten Neue Komponenten (Rumpf, Flügel) werden mit vorgegebenen Standardwerten erzeugt, die über die Editierfunktionen angepasst werden können. Beim Erstellen wird zunächst nur ein Segment erstellt. Weitere Segmente können bei Bedarf hinzugefügt (an das bestehende Segment angehängt) werden.

Editieren Die Funktionen zum Editieren machen den Großteil der Anforderungen aus und werden daher einzeln aufgeführt (siehe unten). Eine Zusammenfassung der Anforderungen zum Editieren der Geometrien ist weiter unten aufgeführt.

Löschen Erstellte Geometrien können wieder gelöscht werden. Es können ganze Entwürfe, Flügelkomponenten und Rumpfkomponenten, aber auch nur einzelne Flügel- oder Rumpfsegmente entfernt werden.

Grafische Ansichten Um die Entwurfsgeometrie grafisch darzustellen, werden drei zweidimensionale Ansichten verwendet. Die Ansichten zeigen den Entwurf von Oben (XY), Vorne (YZ) und einer Seite (XZ). Die Geometrien werden, wie in Abbildung 3.1 gezeigt, in stark vereinfachter Form dargestellt. In den Ansichten werden die in Tabelle 3.1 aufgeführten Elemente dargestellt. Die gezeichneten Größen werden bei Anpassungen der Parameter live aktualisiert. D.h. bei der Verwendung von Schiebereglern sind die Änderungen direkt in den Ansichten sichtbar.

Um bestehende Flugzeugentwürfe anhand von Zeichnungen nachzumodellieren gibt es die Möglichkeit Hintergrundbilder in die Ansichten zu laden. Die Hintergrundbilder können vom Benutzer skaliert und verschoben werden um mit den gezeichneten Linien der Ansichten zu überlappen.

TIGL-Viewer Zusätzlich zu den vereinfachten Ansichten gibt es die Möglichkeit über das Menü des SGG eine lokal installierte Version des TIGL-Viewers zu starten. Für den Datenaustausch zwischen SGG und dem TIGL-Viewer wird eine extra für diesen Zweck angelegte CPACS-Datei verwendet, welche nur die benötigten Geometrien beinhaltet. Auf diese Weise werden Verzögerungen, die durch einen Datenoverhead auftreten können vermieden.

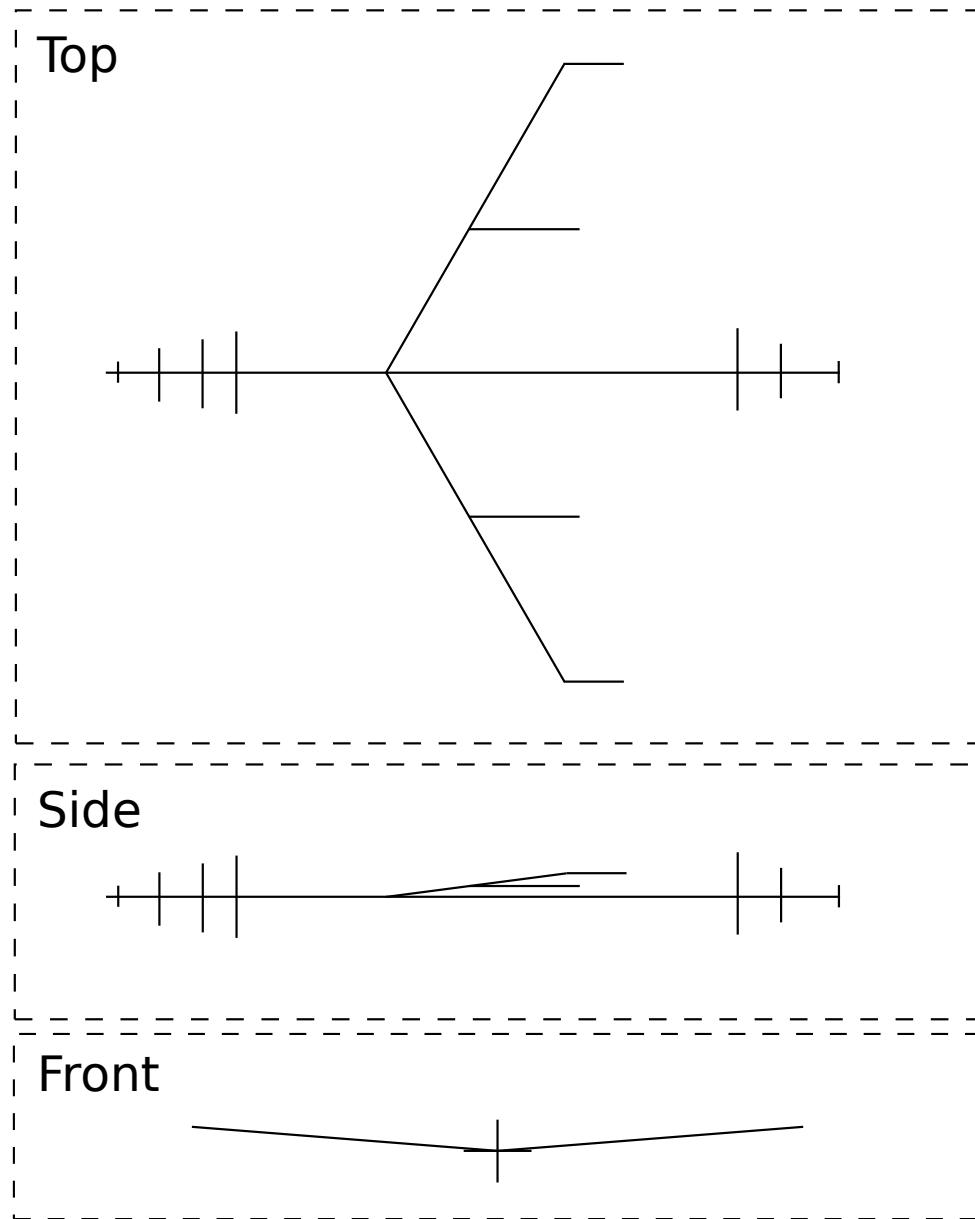


Abbildung 3.1: Grafische Ansichten

Die Anforderungen zum Editieren der Geometrien werden wie folgt zusammengefasst:

Komponententransformation Für komplett Komponenten (Rumpf, Flügel) kann die Position, Orientierung und Größe (Skalierung) geändert werden.

Rumpflinien	die Verbindungslien zwischen zwei Sektionen für ein Segment
Rumpfsektionen	Höhe und Breite der Rumpfsektionen sowie deren Orientierung
Tragflächenline	die Verbindungslien zwischen zwei Sektionen für ein Segment (kann Vorderkante, Hinterkante oder etwas dazwischen sein)
Tragflächensektionen	Höhe und Breite der Tragflächensektionen sowie deren Orientierung (Verwindung)

Tabelle 3.1: Elemente der Ansichten

Komponentenattribute Attribute der Komponenten wie Name, uID, Symmetrie und Symmetrieebene können bearbeitet werden. Außerdem kann für eine Komponente angegeben werden, ob diese einer anderen Komponente untergeordnet werden soll („parent“ Attribut).

„sections“ transformieren Die einzelnen „section“-Knoten können transformiert (verschoben, rotiert, skaliert) werden. Dies gilt sowohl für die Rumpf- als auch für die Flügelsektionen.

„elements“ transformieren Die „elements“, die den CPACS-„section“-Knoten untergeordnet sind können ebenfalls transformiert (verschoben, rotiert, skaliert) werden. Dies gilt sowohl für die Rumpf- als auch für die Flügelemente.

Rumpfquerschnitte auswählen Als Querschnittsformen können bereits in der geladenen CPACS-Datei bestehende Profile verwendet werden. Zusätzlich stehen in einer dem SGG beiliegenden Bibliothek Standardprofile zur Verfügung.

Flügelprofile auswählen Flügelprofile können ebenfalls aus der aktuellen CPACS-Datei sowie aus einer beiliegenden Bibliothek ausgewählt werden.

Rumpfsegmentparameter Ein Rumpfsegment wird von zwei Rumpfquerschnitten aufgespannt. Zur Definition der Rumpfsegmente werden die in Tabelle 3.2 aufgeführten Parameter in den angegebenen Kombinationen verwendet. Die Parameter der Pfeilung, V-Stellung und Länge des Segmentes werden direkt aus der Definition des „positioning“-Knotens in CPACS verwendet (siehe Abschnitt 3.2). Da ein Rumpfsegment aus zwei Querschnitten besteht, die unterschiedliche Breiten und Höhen haben können, werden hier die Werte des zweiten Querschnittes verwendet. In Parameterkombinationen in denen die Höhe nicht angegeben wird, ergibt sich diese aus der Breite indem das Verhältnis von Höhe zu Breite konstant bleibt.

Flügelsegmentparameter Die Geometrie für Flügelsegmente wird ähnlich wie die für Rumpfsegmente definiert. Flügelsegmente werden ebenfalls von zwei Querschnit-

ten aufgespannt, wobei das zweite über die im „positioning“-Element beschriebenen Parameter positioniert wird. Für die Konfiguration des Flügelsegments werden neben der Pfeilung, der V-Stellung und der Spannweite weitere typische Entwurfsparameter bereitgestellt, welche in verschiedenen Kombinationen verwendet werden können. Die unterschiedlichen Parameterkombinationen sind in Tabelle 3.3 dargestellt.

Parameter	Kombinationen			
	Pfeilung φ	X	X	X
	V-Stellung Ψ	X	X	X
	Länge des Segmentes L	X	X	-
	Breite B	X	X	-
	Höhe H	-	X	-
	Breiten zu Längen Verhältnis B/L	-	-	X

Tabelle 3.2: Parameterkombinationen Rumpfsegment

Parameter	Kombinationen			
	Pfeilung φ	X	X	X
	V-Stellung Ψ	X	X	X
	Profilverwindung ε	X	X	X
	Spannweite $span$	X	-	-
	Zuspitzung $\tau = \frac{t_a}{t_i}$	X	X	X
	Streckung $\Lambda = \frac{span^2}{S_{ref}}$	-	X	-
	Fläche des Segmentes S_{ref}	-	-	X

Tabelle 3.3: Parameterkombinationen Flügelsegment

Die Kernfunktionalität des Simple Geometry Generators ist der vereinfachte Zugriff auf Komponententeile über intuitive Geometrieparameter. Diese Funktionalität spiegelt sich in den Anforderungen „Rumpfsegmentparameter“ und „Flügelsegmentparameter“ wieder.

Um basierend auf CPACS-Daten intuitive Parameter bereitzustellen werden im SGG „Parts“ verwendet. Parts können prinzipiell auch als Segmente betrachtet werden. Da der „Segment“-Begriff aber bereits in CPACS definiert ist, wird in dieser Arbeit der „Part“-Begriff verwendet um zwischen der CPACS und der SGG Definition zu unterscheiden. Die „Rumpfsegmentparameter“ sind im FuselagePart und die „Flügelsegmentparameter“ im WingPart implementiert. Um die Definition der Parts zu verstehen ist eine grobe Kenntnis über die verwendete CPACS-Struktur notwendig. Daher werden im folgenden Abschnitt zunächst Teile der CPACS-Struktur vorgestellt und in Abschnitt 3.3 dann der Umgang mit Geometrien beschrieben, der die Berechnung der Part-Parameter beinhaltet.

3.2 CPACS

Das Common Parametric Aircraft Configuration Schema (CPACS) ist eine XML-Schnittstelle die seit 2005 am DLR entwickelt wird. Ein XML-Dokument besteht aus hierarchisch (in einer Baumstruktur) angeordneten Daten. Es beinhaltet ein Wurzelement dem alle anderen Elemente untergeordnet sind. Um die Verlinkungen von Teilbäumen zu ermöglichen, werden in CPACS uIDs (Unique Identifier) verwendet. Die uID ist eine eindeutige Kennung für die unterschiedlichen Elemente. Die Definition von CPACS, welche in einer XML-Schemadatei angegeben ist, legt die allgemeine Struktur von CPACS-Dateien fest. Fünf Jahre lang wurde CPACS entwickelt, bis im Jahr 2010 die erste CPACS Version 1.0 veröffentlicht wurde. Parallel zu CPACS sind auch Softwaretools entwickelt worden, die CPACS-Datensätze verwenden. Dazu gehören z.B. Tools wie TIGLViewer, VAMPzero und Chameleon/RCE. Im März 2012 wurde die CPACS Version 2.0 zusammen mit VAMPzero und Chameleon/RCE veröffentlicht.

Geometrien in CPACS sind durch zueinander positionierten Querschnitten aufgebaut. Durch die Verbindung dieser Querschnitte werden die Volumenkörper definiert. Da der Kern dieser Arbeit den Umgang mit CPACS-Geometrien beinhaltet, ist eine detailliertere Kenntnis der CPACS-Geometrien unerlässlich. Daher wird im Folgenden kurz die Struktur der CPACS-Geometrien vorgestellt. Für eine detaillierte Beschreibung der CPACS-Geometrien wird an dieser Stelle auf die CPACS-Dokumentation [Deu12] verwiesen.

Das Wurzelement für eine CPACS-Datei ist „cpacs“. Dem „cpacs“-Element sind verschiedene Knoten untergeordnet. Das „cpacs“-Element mit den beiden für diese Arbeit relevanten Elementen ist in Abbildung 3.2 dargestellt. Das „head“-Element umfasst all-

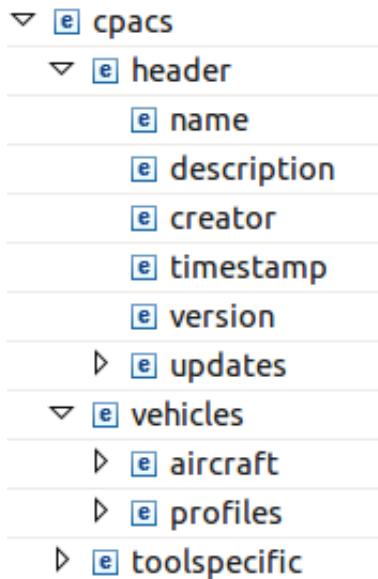


Abbildung 3.2: „cpacs“-Element

gemeine Daten zum Dokument wie den Namen, eine Beschreibung, Informationen zum Ersteller, den Zeitstempel der Erstellung, eine Versionsangabe, sowie Aktualisierungs-

Informationen. Unter dem „vehicles“-Element befinden sich die Unterelemente „aircraft“ in dem beliebig viele Flugzeugentwürfe abgelegt werden können, und „profiles“ welches eine Sammlung von Profilen beinhaltet. „profiles“ enthält z.B. die Rumpf- und Tragflächenprofile, die von den Querschnitten („element“-Elementen) verwendet werden, siehe Abbildung 3.3. Die Profile sind wie in Abbildung 3.4 dargestellt abgelegt. Die „x“-, „y“-



Abbildung 3.3: „profiles“-Element

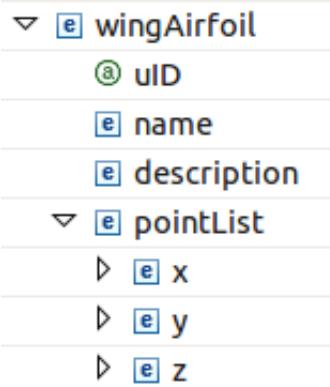
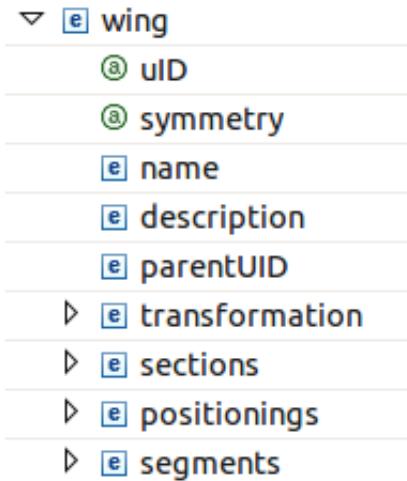


Abbildung 3.4: „wingAirfoil“-Element

und „z“-Tags beinhalten Listen der jeweiligen Punktkoordinaten. Da die „fuselageProfiles“ analog zu den „wingAirfoils“ aufgebaut sind, werden diese hier nicht gesondert aufgeführt.

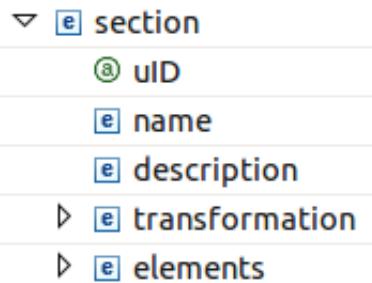
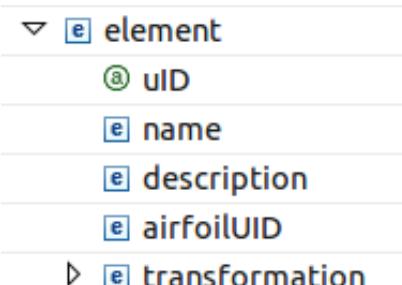
Auch die „wing“- und „fuselage“-Elemente sind fast identisch aufgebaut, daher wird hier lediglich die Struktur des „wing“-Elementes erläutert und wo notwendig auf Unterschiede zum „fuselage“-Element hingewiesen. Abbildung 3.5 zeigt die Struktur des „wing“-Elementes. Wie bereits erwähnt, sind Geometrien in CPACS durch zueinander positionierte Querschnitte aufgebaut. Dies spiegelt sich im „wing“-Element wieder. Die Querschnitte sind in dem „sections“-Element abgelegt. Deren Positionierung erfolgt unter dem „positionings“-Element. Welche „element“-Elemente zu einem Volumenkörper verbunden werden wird im „segments“-Element festgelegt.

Um die globalen Koordinaten von Geometriepunkten zu Berechnen, wie es für die Ansichten notwendig ist (siehe Abschnitt 3.3.2), wird im Folgenden der Aufbau der Geometrien detailliert beschrieben. Die „section“-Elemente (Abbildung 3.6) sind wiederum aus einem oder mehr „element“-Elementen (Abbildung 3.7) aufgebaut, welche auf unter „profiles“ abgelegte Profile verweisen. Die „wing“, „fuselage“, „section“- und „element“-Elemente beinhalten jeweils ein „transformation“-Element, das die Transformationen in dem jeweiligen Koordinatensystem festlegt. Abbildung 3.8 zeigt die drei

**Abbildung 3.5:** „wing“-Element

Transformationen (Translation, Rotation, Skalierung).

Die Transformationen des „element“-Knotens werden auf das referenzierte Profil an-

**Abbildung 3.6:** „section“-Element**Abbildung 3.7:** „element“-Element

gewandt. Dabei wird zunächst skaliert, dann um den Ursprung ([0,0,0]) rotiert und als letztes verschoben. Der Referenzpunkt für die Transformationen des „section“-Elementes ist der Ursprung des jeweiligen „element“-Knotens. Die Skalierung jedoch bildet eine Ausnahme, da diese, wie auch die Skalierung des „element“-Knotens im „element“-Koordinatensystem durchgeführt wird. Eine Rotation in der „element“-Transformation hat somit keinen Einfluss auf die Skalierung im „section“-Knoten. Diese Besonderheit

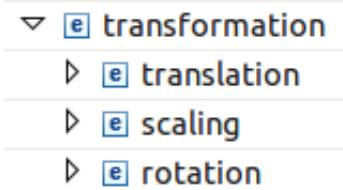


Abbildung 3.8: „transformation“-Element

macht es notwendig die Skalierung des „section“-Elementes noch vor der Rotation der „element“-Transformation auszuführen und wird bei der Berechnung der Koordinaten in Abschnitt 3.3.2 berücksichtigt. Die Transformation der „wing“- und „fuselage“-Elemente werden wiederum in der gleichen Reihenfolge wie im „element“-Knoten angewendet. Diese Transformationen auf Querschnittspunkte angewendet liefert deren Koordinaten im jeweiligen Komponenten-Koordinatensystem (hier „wing“-KOS). Um die Koordinaten im globalen Koordinatensystem zu berechnen ist noch das „parentUID“-Attribut der Komponente zu berücksichtigen. Mittels „parentUID“-Attribut können die Positionen von Komponenten relativ zu anderen Komponenten (über deren uID verlinkt) angegeben werden. Das bedeutet, dass die Translation der „parent“-Komponente zu der eigenen Translation zu addieren ist. Wenn eine mittels „parentUID“ referenzierte Komponente ebenfalls einer weiteren Komponente untergeordnet ist, sind alle auf diesem Wege folgenden Translationen zu addieren. Die Rotation und Skalierung der referenzierten Komponenten haben keinen Einfluss auf die referenzierende Komponente. Ein Beispiel für die Berechnung von Koordinaten befindet sich in Abschnitt 3.3.2.

3.2.1 CPACS-Pfad

Um den Zugriff auf CPACS-Knoten zu vereinfachen, wird eine Funktion definiert, welche den Zugriff durch Angabe eines CPACS-Pfades ermöglicht. Der CPACS-Pfad wird im SGG verwendet um Elemente im XML-Dokument zu platzieren.

Die CPACS-Pfade werden nach folgendem Muster erstellt:

- Der CPACS-Pfad ist wie ein Unix-Verzeichnisbaum aufgebaut. Die einzelnen Ebenen werden durch ein „/“ voneinander getrennt.
- Wenn ein Knoten mehrere Elemente desselben Typs beinhaltet, erfolgt ein eindeutiger Zugriff über die uID (z.B. aircraft/model[Aircraft1] => uID = 'Aircraft1').

Um die uIDs eindeutig zu machen, wurden in dieser Arbeit die folgenden Konventionen verwendet:

- Die uIDs sind nur in Elementen vorhanden, deren Name an einem bestimmten Ort nicht eindeutig ist (z.B. wings/wing[Wing1]).
- Die Bezeichnungen für die uIDs beginnen beim ersten, nicht eindeutigen Element auf dem CPACS-Pfad (z.B. aircraft/model[Aircraft1] => uID = 'Aircraft1').

- In der uID werden die unterschiedlichen Ebenen durch einen Unterstrich ‘_’ voneinander getrennt (’/cpacs/vehicles/aircraft/model[Aircraft1]/wings/wing[Wing1]’ => uID = ’model_wing’).
- Für jedes Element wird ein eindeutiges Kürzel festgelegt, das in der uID vermerkt wird. In Tabelle 3.4 sind die Kürzel aufgeführt die vom SGG beim Erstellen neuer Geometrien verwendet werden.

Tabelle 3.4: uIDs von Elementen

Element	uID
model	Aircraft
wing	Wing
fuselage	Fuse
section	Sec
element	Elem
segment	Seg
positioning	Pos

Die Funktion „getObjFromXPath()“ liefert das zu einem CPACS-Pfad gehörende CPACS-Objekt aus der CPACS-Struktur zurück. Sollte das durch den Pfad angegebene Objekt nicht existieren, wird es erstellt. Dabei werden alle Knoten bis zu dem Objekt ebenfalls erstellt. Sollten mehrere Knoten eines Typs vorhanden, aber keine uID angegeben sein, so wird der erste Knoten verwendet.

3.3 Umgang mit Geometrien

Dieser Abschnitt erläutert Vorgänge zum Umgang mit Geometrien. Zunächst wird erläutert, wie die Parts (WingPart/FuselagePart) definiert sind, wie die Parameter berechnet werden und welchen Einfluss sie auf die CPACS-Daten haben. Danach wird beschrieben, welche Schritte zum Berechnen der Koordinaten für die Ansichten durchgeführt werden. Abschließend wird erläutert, welche Zusammenhänge beim Erstellen und Löschen von Geometrien im SGG bestehen und die Annahmen die dabei getroffen werden aufgeführt.

3.3.1 Parts

Um die Erstellung neuer Geometrien so einfach wie möglich zu halten werden im SGG „Parts“ verwendet, welche die für CPACS verwendeten Strukturen (sections, elements, positionings, segments) zu einem intuitiven Teil (Part) verbinden. Diese Maßnahme vereinfacht die Erstellung von Geometrien erheblich. Bei der Verwendung von Parts werden außerdem immer konsistente/valide CPACS-Daten erzeugt. Um nicht an Flexibilität einzubüßen, können die CPACS-Strukturen auch direkt bearbeitet werden. Beim Erstellen oder Entfernen von CPACS-Elementen ist es allerdings dem Benutzer selbst überlassen sich der Sinnhaftigkeit der erstellten Geometrien zu überzeugen.

Festgelegt wird ein Part durch ein „section“-element. Die Positionierung bezüglich der vorherigen „section“, sowie die Größe und Orientierung legt dann die Form des Parts fest. Die Form wird immer durch den ersten „element“-Knoten innerhalb der „section“ bestimmt. Da die Reihenfolge der Elemente in XML generell beliebig ist, sollte, bei der Verwendung mehrerer „element“-Knoten innerhalb einer „section“, das verwendete „element“ überprüft werden.

3.3.1.1 Berechnen der WingPart-Parameter

Dieser Abschnitt beschreibt, wie die WingPart-Parameter berechnet und in CPACS-Parameter überführt werden. Um die Parts auf unterschiedliche Weisen definieren zu können sind vier verschiedene Parameterkombinationen definiert, zwischen denen gewählt werden kann. Die WingParts werden von insgesamt fünf Parametern definiert, deren Kombinationen in Tabelle 3.3 aus Abschnitt 3.1 gezeigt werden. Der Einfluss einiger Parameter auf die CPACS-Daten variiert je nach Parameterkombination, da je nach Kombination unterschiedliche Parameter festgesetzt werden. Für diese Parameter werden alle Gleichungen für die verschiedenen Berechnungen aufgeführt. Je nach Parameterkombination werden unterschiedliche Berechnungen durchgeführt, die im Folgenden vorgestellt werden. Abbildung 3.9 zeigt die Bedeutung der verwendeten Parameter. Für die Berech-

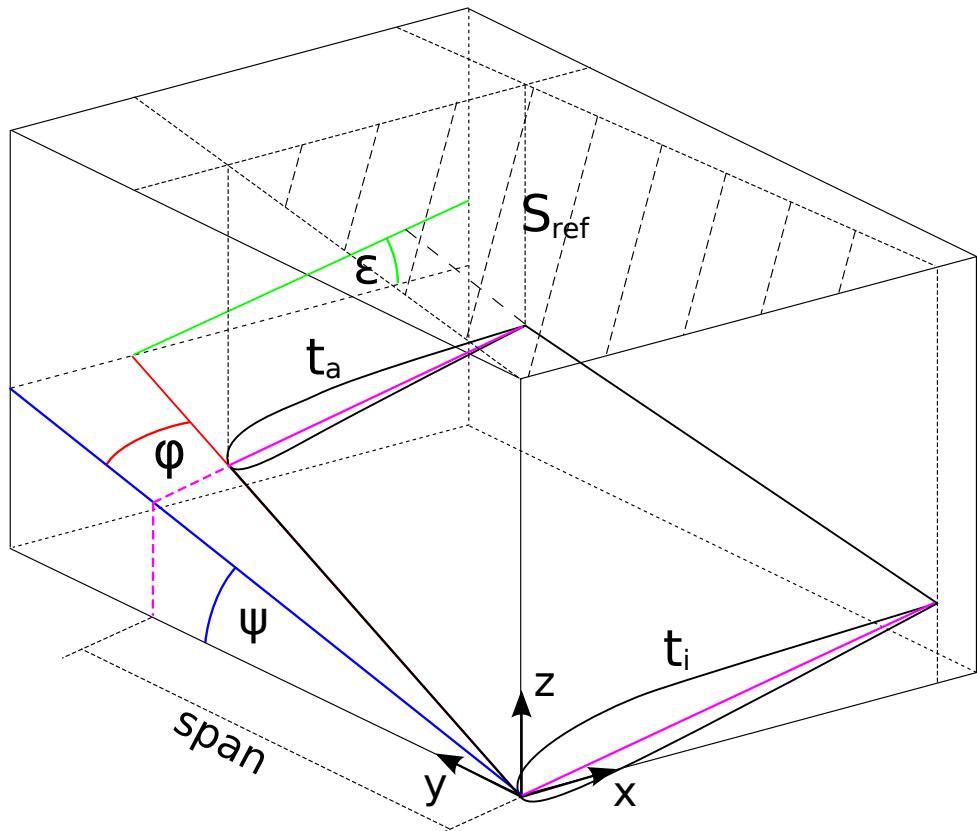


Abbildung 3.9: Wingpart-Parameter

nung der Part-Parameter werden die ersten „element“-Knoten der „section“-Elemente

verwendet. Zur Kontrolle wird das zur Berechnung der Parameter verwendete „segment“ im WingPartWidget angezeigt (siehe Abschnitt 3.5.6).

Die Parameter Pfeilung (Sweep), V-Stellung (Dihedral) werden ohne Umrechnung verwendet, da diese im SGG und CPACS die gleichen Größen beschreiben.

Die Spannweite *span* wird aus der im CPACS-„positioning“ angegebenen Länge *L*, sowie der Pfeilung φ und der V-Stellung ψ berechnet. Gleichung 3.1 zeigt die Formel zur Berechnung der Spannweite aus den genannten Größen.

$$span = L \cdot \cos(\varphi) \cdot \cos(\psi) \quad (3.1)$$

Um die Länge *L* aus einer gegebenen Spannweite und gegebener Pfeilung und V-Stellung zu berechnen, wird Gleichung 3.1 nach *L* umgeformt und ergibt Gleichung 3.2:

$$L = \frac{span}{\cos(\varphi) \cdot \cos(\psi)} \quad (3.2)$$

Für Parts von symmetrischen Komponenten wird die Spannweite automatisch verdoppelt. Das bedeutet, dass die Gleichung 3.1 zu 3.3 und Gleichung 3.2 zu 3.4 wird.

$$span = 2 \cdot L \cdot \cos(\varphi) \cdot \cos(\psi) \quad (3.3)$$

$$L = \frac{span}{2 \cdot \cos(\varphi) \cdot \cos(\psi)} \quad (3.4)$$

Der „Twist“ beschreibt die geometrische Profilverwindung des Parts. Er gibt den Winkel um die y-Achse vom „toElement“ bezüglich dem „fromElement“ an. Gleichung 3.5 zeigt wie der „Twist“ $d\varepsilon$ aus dem Anstellwinkel des ersten „element“-Knotens der „fromSection“ ε_{from} und der „toSection“ ε_{to} berechnet wird.

$$d\varepsilon = \varepsilon_{to} - \varepsilon_{from} \quad (3.5)$$

Der Anstellwinkel eines „element“-Knotens berechnet sich, wie Gleichung 3.6 zeigt, als Summe der „section“- ε_{sec} und „element“-Rotation ε_{elem} um die y-Achse.

$$\varepsilon = \varepsilon_{sec} + \varepsilon_{elem} \quad (3.6)$$

Durch Einsetzen von Gleichung 3.6 in 3.5 ergibt sich 3.7.

$$d\varepsilon = (\varepsilon_{to,sec} + \varepsilon_{to,elem}) - (\varepsilon_{from,sec} + \varepsilon_{from,elem}) \quad (3.7)$$

Um die geometrische Profilverwindung in CPACS-Parameter zu überführen, wird die Gleichung 3.7 zu 3.8 umgeformt.

$$\varepsilon_{to,elem} = d\varepsilon + (\varepsilon_{from,sec} + \varepsilon_{from,elem}) - \varepsilon_{to,sec} \quad (3.8)$$

Gleichung 3.8 zeigt, dass lediglich die Rotation des „toElement“-Knotens aus der geforderten Profilverwindung berechnet wird. Die Rotation der „toSection“-, „fromSection“- und „fromElement“ Knoten bleibt unverändert. Das „TaperRatio“ ist die Zuspitzung des Parts. Die Zuspitzung beschreibt das Verhältnis der Profiltiefen der beiden Querschnitte, siehe Gleichung 3.9.

$$\tau = \frac{t_{to}}{t_{from}} \quad (3.9)$$

Die Profiltiefen berechnen sich wie in Gleichung 3.10 gezeigt, aus den Skalierungsgrößen in Profilachsen-Richtung (in der Regel die x-Achse) und der normierten Profiltiefe von 1m.

$$t = \hat{t}_{sec} \cdot \hat{t}_{elem} \cdot 1m \quad (3.10)$$

Ein Einsetzen von Gleichung 3.10 in 3.9 ergibt 3.11.

$$\tau = \frac{\hat{t}_{to,sec} \cdot \hat{t}_{to,elem}}{\hat{t}_{from,sec} \cdot \hat{t}_{from,elem}} \quad (3.11)$$

Der Einfluss der Zuspitzung auf die CPACS-Parameter wird begrenzt sich auf die Skalierung des „toElement“-Knotens. Diese berechnet sich anhand von Gleichung 3.12.

$$\hat{t}_{to,elem} = \tau \cdot \frac{\hat{t}_{from,sec} \cdot \hat{t}_{from,elem}}{\hat{t}_{to,sec}} \quad (3.12)$$

Um die Form des Querschnittes nicht zu verändern, wird die Profildicke ebenfalls skaliert so dass das Verhältnis von Profildicke \hat{d}_{elem} zu Profillänge \hat{t}_{elem} gleich bleibt. Dazu wird zunächst das in CPACS abgelegte Verhältnis d/t berechnet, siehe Gleichung 3.13.

$$d/t = \frac{\hat{d}_{to,elem}^*}{\hat{t}_{to,elem}^*} \quad (3.13)$$

Wobei $\hat{d}_{to,elem}^*$ und $\hat{t}_{to,elem}^*$ die alte/vorherige Profildicke und -länge des „toElement“-Knotens in CPACS sind.

Um den neuen Skalierungsfaktor für die Profildicke zu berechnen, wird Gleichung 3.14 verwendet.

$$\hat{d}_{to,elem} = d/t \cdot \hat{t}_{to,elem} = \frac{\hat{d}_{to,elem}^*}{\hat{t}_{to,elem}^*} \cdot \hat{t}_{to,elem} \quad (3.14)$$

Die Referenzfläche S_{ref} ist die in die XY-Ebene projizierte Fläche des Flügel-Parts. Sie kann anhand von Gleichung 3.15 berechnet werden.

$$S_{ref} = \frac{t_{from} \cdot \cos(\varepsilon_{from}) + t_{to} \cdot \cos(\varepsilon_{to})}{2} \cdot span \quad (3.15)$$

Da die geometrische Profilverwindung in der Regel sehr klein ist ($d\varepsilon < 15^\circ$) wird deren Einfluss auf die Referenzfläche für die Berechnung im SGG vernachlässigt. Im SGG wird daher Gleichung 3.16 verwendet.

$$S_{ref} = \frac{t_{from} + t_{to}}{2} \cdot span \quad (3.16)$$

Mit den Gleichungen 3.10 und 3.1 lässt sich Gleichung 3.16 komplett durch CPACS-Parameter darstellen. Das Ergebnis zeigt Gleichung 3.17.

$$S_{ref} = \frac{\hat{t}_{from,sec} \cdot \hat{t}_{from,elem} \cdot 1m + \hat{t}_{to,sec} \cdot \hat{t}_{to,elem} \cdot 1m}{2} \cdot L \cdot \cos(\varphi) \cdot \cos(\psi) \quad (3.17)$$

Für symmetrische Flügel wird, die Spannweite und somit auch die Referenzfläche verdoppelt (Gleichung 3.18).

$$S_{ref} = (\hat{t}_{from,sec} \cdot \hat{t}_{from,elem} \cdot 1m + \hat{t}_{to,sec} \cdot \hat{t}_{to,elem} \cdot 1m) \cdot L \cdot \cos(\varphi) \cdot \cos(\psi) \quad (3.18)$$

Die Referenzfläche ist Teil der Parameterkombinationen SetB und SetD (siehe Tabelle 3.3). In SetB legt die Referenzfläche die Zuspitzung und somit die Skalierung des „toElement“-Knotens fest. Durch das Auflösen von Gleichung 3.17 bzw. 3.18 nach $\hat{t}_{to,elem}$ ergibt sich der Skalierungsfaktor aus Gleichung 3.19 bzw. 3.20.

$$\hat{t}_{to,elem} = \frac{\left(\frac{2 \cdot S_{ref}}{L \cdot \cos(\varphi) \cdot \cos(\psi)} - \hat{t}_{from,sec} \cdot \hat{t}_{from,elem} \cdot 1m \right)}{\hat{t}_{to,sec} \cdot 1m} \quad (3.19)$$

$$\hat{t}_{to,elem} = \frac{\left(\frac{S_{ref}}{L \cdot \cos(\varphi) \cdot \cos(\psi)} - \hat{t}_{from,sec} \cdot \hat{t}_{from,elem} \cdot 1m \right)}{\hat{t}_{to,sec} \cdot 1m} \quad (3.20)$$

Die Profildicke $\hat{d}_{to,elem}$ wird, wie schon für die Zuspitzung, durch Gleichung 3.14 berechnet. In SetD wird die Zuspitzung durch den Benutzer festgelegt und die Referenzfläche beeinflusst die Spannweite bzw. in CPACS den L -Parameter (siehe Gleichung 3.1). Durch das Auflösen von Gleichung 3.17 bzw. 3.18 nach L ergibt sich Gleichung 3.21 bzw. für symmetrische Flügel Gleichung 3.22.

$$L = \frac{2 \cdot S_{ref}}{\cos(\varphi) \cdot \cos(\psi) \cdot (\hat{t}_{from,sec} \cdot \hat{t}_{from,elem} \cdot 1m + \hat{t}_{to,sec} \cdot \hat{t}_{to,elem} \cdot 1m)} \quad (3.21)$$

$$L = \frac{S_{ref}}{\cos(\varphi) \cdot \cos(\psi) \cdot (\hat{t}_{from,sec} \cdot \hat{t}_{from,elem} \cdot 1m + \hat{t}_{to,sec} \cdot \hat{t}_{to,elem} \cdot 1m)} \quad (3.22)$$

Das „Aspect Ratio“ ist die Streckung des Flügel-Parts. Sie wird, wie Gleichung 3.23 zeigt, aus der Referenzfläche und der Spannweite berechnet.

$$\Lambda = \frac{S_{ref}}{span^2} \quad (3.23)$$

Durch Einsetzen von 3.1 und 3.17 bzw. für einen symmetrischen Flügel 3.3 und 3.18 in Gleichung 3.23 ergibt sich die Streckung Λ aus Gleichung 3.24 bzw. 3.25.

$$\Lambda = \frac{\hat{t}_{from,sec} \cdot \hat{t}_{from,elem} \cdot 1m + \hat{t}_{to,sec} \cdot \hat{t}_{to,elem} \cdot 1m}{2 \cdot L \cdot \cos(\varphi) \cdot \cos(\psi)} \quad (3.24)$$

$$\Lambda = \frac{\hat{t}_{from,sec} \cdot \hat{t}_{from,elem} \cdot 1m + \hat{t}_{to,sec} \cdot \hat{t}_{to,elem} \cdot 1m}{4 \cdot L \cdot \cos(\varphi) \cdot \cos(\psi)} \quad (3.25)$$

Die Verwendung der Streckung zur Definition der Geometrie nur in SetC möglich. In diesem Fall legt die Streckung, in Analogie zur Zuspitzung, die Profiltiefe des „toElement“-Knotens fest. Dazu wird Gleichung 3.24 bzw. für symmetrische Flügel Gleichung 3.25 nach $\hat{t}_{to,elem}$ aufgelöst. Dadurch entstehen Gleichung 3.26 bzw. 3.27.

$$\hat{t}_{to,elem} = \frac{2 \cdot L \cdot \cos(\varphi) \cdot \cos(\psi) \cdot \Lambda - (\hat{t}_{from,sec} \cdot \hat{t}_{from,elem} \cdot 1m)}{\hat{t}_{to,sec} \cdot 1m} \quad (3.26)$$

$$\hat{t}_{to,elem} = \frac{4 \cdot L \cdot \cos(\varphi) \cdot \cos(\psi) \cdot \Lambda - (\hat{t}_{from,sec} \cdot \hat{t}_{from,elem} \cdot 1m)}{\hat{t}_{to,sec} \cdot 1m} \quad (3.27)$$

Auch hier wird Gleichung 3.14 verwendet um parallel zur Profiltiefe die Profildicke $\hat{d}_{to,elem}$ anzupassen und die Querschnittsform zu erhalten.

3.3.1.2 Berechnen der FuselagePart-Parameter

Für die Fuselageteile werden die folgenden Parameter verwendet:

- Pfeilung
- V-Stellung
- Länge des Parts
- Breite des Rumpfquerschnitts
- Höhe des Rumpfquerschnitts
- Das Verhältnis von Breite zu Länge (B/L)

Die Pfeilung und V-Stellung des FuselageParts sind analog zu denen des WingParts definiert. Sie werden direkt im zugehörigen „positioning“-Knoten abgelegt. Für den

FuselagePart wird nicht, wie bei der Spannweite eine projizierte Länge, sondern die komplette Länge des Parts angegeben, wie sie in dem „positioning“-Element definiert ist. Das heißt auch, dass die Längenangabe nicht durch die Symmetrie der Komponente beeinflusst wird.

Die Breite („Width“) und Höhe („Height“) eines FuselageParts legen die Breite und Höhe der „toElements“ fest. Daher können die Werte über die Partgeometrie duchaus variieren. Die Breite und Höhe werden zunächst aus dem im „toElement“ referenzierten Profil gelesen und dann anhand der Skalierungsfaktoren des „element“- und „section“-Knotens skaliert. Mit der originalen Profilbreite B_{prof} und Profilhöhe H_{prof} lassen sich die Gesamtbreite und Gesamthöhe anhand von Gleichung 3.28 bzw. 3.29 berechnen.

$$B_{part} = B_{prof} \cdot \hat{B}_{to,sec} \cdot \hat{B}_{to,elem} \quad (3.28)$$

$$H_{part} = H_{prof} \cdot \hat{H}_{to,sec} \cdot \hat{H}_{to,elem} \quad (3.29)$$

Wobei $\hat{B}_{to,sec}$ und $\hat{B}_{to,elem}$ die Skalierungsfaktoren für die Breite, und $\hat{H}_{to,sec}$ und $\hat{H}_{to,elem}$ die Skalierungsfaktoren für die Höhe der „toSection“ bzw. des „toElements“ sind.

Die Breite und Höhe des Parts beeinflussen die Skalierungsfaktoren des „toElement“-Knotens. Durch Umformen der Gleichungen 3.28 und 3.29 nach $\hat{B}_{to,elem}$ bzw. $\hat{H}_{to,elem}$ ergeben sich die Gleichungen 3.30 und 3.31.

$$\hat{B}_{to,elem} = \frac{B_{part}}{B_{prof} \cdot \hat{B}_{to,sec}} \quad (3.30)$$

$$\hat{H}_{to,elem} = \frac{H_{part}}{H_{prof} \cdot \hat{H}_{to,sec}} \quad (3.31)$$

B/L beschreibt das Verhältnis zwischen der Breite B und der Länge des Rumpfparts L . Mit B_{part} aus Gleichung 3.28 ergibt sich B/L somit anhand von Gleichung 3.32.

$$B/L = \frac{B_{prof} \cdot \hat{B}_{to,sec} \cdot \hat{B}_{to,elem}}{L} \quad (3.32)$$

Je nach gewähltem Parameterset beeinflusst B/L die Länge L oder die Breite $\hat{B}_{to,elem}$ des Parts. Somit gilt für SetD die Gleichung 3.33 und für SetC Gleichung 3.34.

$$L = \frac{B_{prof} \cdot \hat{B}_{to,sec} \cdot \hat{B}_{to,elem}}{B/L} \quad (3.33)$$

$$\hat{B}_{to,elem} = \frac{B/L \cdot L}{B_{prof} \cdot \hat{B}_{to,sec}} \quad (3.34)$$

Bei der Verwendung von SetA, SetC und SetD wird die Höhe des Rumpfes in Abhängigkeit der Breite skaliert um die Querschnittsform nicht zu verändern. Dazu wird zunächst das Verhältnis von Höhe $\hat{H}_{to,elem}^*$ zu Breite $\hat{B}_{to,elem}^*$ vor der Änderung von B/L anhand von Gleichung 3.35 ermittelt.

$$\frac{H}{B} = \frac{\hat{H}_{to,elem}^*}{\hat{B}_{to,elem}^*} \quad (3.35)$$

Bei gleichbleibendem H/B ergibt sich die Höhe $\hat{H}_{to,elem}$ somit aus der Breite nach Gleichung 3.36.

$$\hat{H}_{to,elem} = \frac{H}{B} \cdot \hat{B}_{to,elem} \quad (3.36)$$

3.3.2 Koordinatenberechnung

Um die Flügelkomponenten in den vereinfachten Ansichten darzustellen, werden die Koordinaten der Vorder- und Hinterkanten der jeweiligen Sektionen benötigt. In Abschnitt 3.2 wurde die Struktur der CPACS-Geometrie vorgestellt, welche die Grundlage für die Berechnung der Koordinaten ist. Der Ablauf der Koordinatenberechnung ist in Abbildung 3.10 dargestellt. Die Abbildung zeigt, dass die Koordinatenberechnung mit der

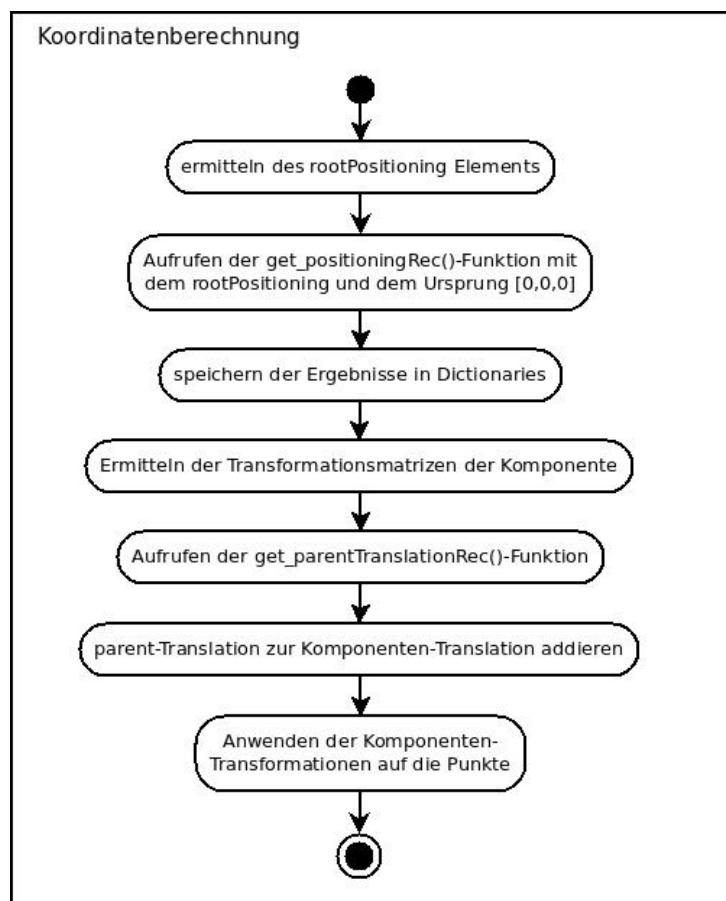


Abbildung 3.10: Koordinatenberechnung

Ermittlung des rootPositioning-Elements beginnt. In dieser Arbeit werden die folgenden zwei Annahmen getroffen:

- Jedes „section“-Element wird durch ein „positioning“-Element positioniert.
- In jeder Komponente gibt es genau eine rootSection.

Aufgrund dieser Annahmen kann die komplette Geometrie über die Abarbeitung der „positioning“-Knoten berechnet werden.

Im zweiten Schritt wird die „get_positioningRec“-Funktion aufgerufen, die Punktkoordinaten der Komponente in den jeweiligen section-Koordinatensystemen zurückliefert. Der Ablauf dieser Funktion wird später erläutert.

Die auf diese Weise ermittelten Koordinaten werden dann noch in das Inertialsystem überführt indem die Komponententransformationen ermittelt und, wie in Gleichung 3.37 beschrieben, auf die Koordinaten angewandt werden.

$$\vec{P}_{transf} = \vec{R}_{sec} \cdot (\vec{R}_{elem} \cdot \vec{S}_{sec} \cdot \vec{S}_{elem} \cdot \vec{P}_{elem} + \vec{T}_{elem}) + \vec{T}_{sec} \quad (3.37)$$

Wobei \vec{P}_{elem} der zu transformierende Punkt (in Profilkordinaten), \vec{P}_{transf} der transformierte Punkt (in Sektionskoordinaten), \vec{S}_{elem} die Skalierungsmatrix des „element“-Knotens, \vec{S}_{sec} die Skalierungsmatrix des „section“-Knotens ist¹ \vec{R}_{elem} die Rotationsmatrix des „element“-Knotens, \vec{R}_{sec} die Rotationsmatrix des „section“-Knotens, \vec{T}_{elem} der Translationsvektor des „element“-Knotens, \vec{T}_{sec} der Translationsvektor des „section“-Knotens ist. Für Komponenten bei denen das „parentUID“-Attribut gesetzt ist, setzt sich die Translation aus der Summe der eigenen Translation und der Translation der referenzierten Komponente zusammen. Sollte die referenzierte Komponente wiederum auf eine weitere Komponente zeigen, ist die Translation rekursiv über die gesamte Kette zu bestimmen. Die Rekursion endet an einer Komponente ohne gesetztes „parentUID“-Attribut.

Die Struktur der „get_positioningRec“-Funktion, welche einen Großteil der Berechnungen durchführt, wird in Abbildung 3.11 gezeigt. Für jedes an diese Funktion übergebene „positioning“-Element wird:

1. zunächst die Ziel-„section“ anhand der „toSectionUID“ aus dem „sections“-Knoten der Komponente herausgesucht.
2. dann wird der durch das „positioning“ festgelegte Vektor bestimmt, der von der „fromSection“ auf die „toSection“ zeigt.

¹ Die Skalierung der „section“-Knoten wird in den gleichen Achsrichtungen ausgeführt wie auch die Skalierung der „element“-Knoten.)

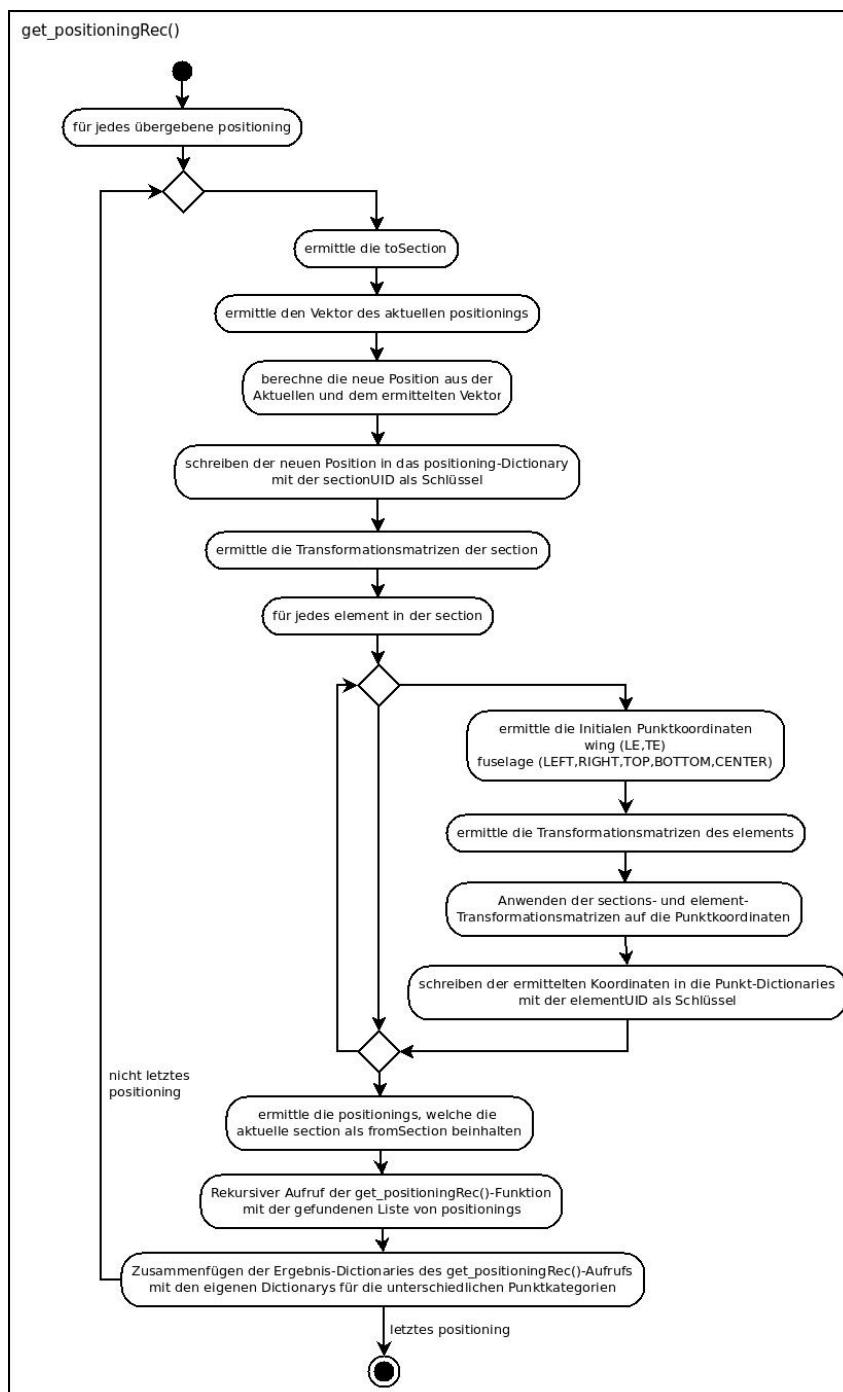


Abbildung 3.11: Struktur der „get_positioningRec“-Funktion

3. Dann wird aus der Position der vorherigen „section“ (als aktuelle Position bezeichnet) und dem ermittelten Vektor die Position der „toSection“ berechnet und in einem Positions-„Dictionary“ mit der „sectionUID“ als Schlüssel gespeichert.
4. Daraufhin werden die Transformationsmatrizen der „section“ bestimmt.

5. Anhand der Transformationsmatrizen werden dann für jedes Element:
 - a) die initialen Punktkoordinaten bestimmt. Für „wingElement“-Knoten werden die „Leading Edge“- und „Trailing Edge“-Koordinaten, für die „fuselageElement“-Knoten werden der Profilmittelpunkt (im Folgenden CENTER genannt), der Linke-Randpunkt auf der horizontalen Achse (LEFT), der Rechte-Randpunkt auf der horizontalen Achse (RIGHT), der Obere-Randpunkt auf der vertikalen Achse (TOP) und der Untere-Randpunkt auf der vertikalen Achse (BOTTOM) bestimmt.
 - b) Die drei Transformationsmatrizen des „element“-Knotens bestimmt.
 - c) Die „section“- und „element“-Transformationen auf die Punktkoordinaten angewandt.
 - d) Und die Ergebnisse der Transformation in die jeweiligen „Dictionarys“ mit den „elementUID“s als Schlüssel geschrieben.
6. Als nächstes werden noch die „positioning“-Elemente ermittelt, die das aktuelle „section“-Element als „fromSectionUID“ referenzieren.
7. Um die folgenden „positioning“-Elemente ebenfalls abzuarbeiten, wird die Funktion „get_positioningRec“ mit den gefundenen „positioning“-Knoten erneut aufgerufen¹.
8. Als letztes werden die Ergebnisse des rekursiven Aufrufs in die in dieser Funktion verwendeten „Dictionarys“ eingetragen und von der Funktion zurückgeliefert.

Da die „wingAirfoils“ als auf ihre Länge normierte Profile abgelegt sind, werden für Querschnitte von „wing“-Komponenten die LE-Koordinate [0,0,0] und die TE-Koordinate [1,0,0] angenommen.

Da die Rumpfprofile auch nicht normiert abgelegt werden, ist es notwendig die Koordinaten dieser vier Punkte für die referenzierten Profile zu ermitteln. Der CENTER-Punkt ist nach Definition der Rumpfquerschnitte immer [0,0,0]. Die Punkte LEFT, RIGHT, TOP und BOTTOM werden anhand der minimalen und maximalen Werte in den jeweiligen Koordinaten des Profils bestimmt. Die Werte ergeben sich wie folgt:

- $LEFT = [0, y_{min}, 0]$
- $RIGHT = [0, y_{max}, 0]$
- $TOP = [0, 0, z_{max}]$
- $BOTTOM = [0, 0, z_{min}]$

¹ Da dieser Aufruf in der Funktion selbst gestartet wird, handelt es sich um einen rekursiven Aufruf.

3.3.3 Erstellen von Geometrien

Dieser Abschnitt erläutert, welche CPACS-Knoten beim Generieren neuer Geometrien erstellt und mit welchen Werten diese initialisiert werden. Erzeugte CPACS-Elemente die ein uID-Attribut besitzen werden beim Erzeugen direkt mit einer generierten uID ausgestattet. Die nach dem in Abschnitt 3.2.1 beschriebenen Muster erstellt wird. Es kann davon ausgegangen werden, dass jedes der erzeugten Elemente eine entsprechende uID zugewiesen bekommt.

Neuer Flugzeugentwurf Beim Erstellen eines neuen Flugzeugentwurfs wird ein Objekt vom Typ aircraftModelTypeSub erstellt und mit einem Namen, einer uID und einer Beschreibung versehen. Außerdem enthält der Entwurf bereits eine initialisierte Tragflächenkomponente. Welche Elemente die Komponente enthält wird im nächsten Absatz beschrieben.

Neue Komponente Die Komponenten werden initialisiert indem ihnen ebenfalls ein Name, eine uID und eine Beschreibung zugewiesen wird. Zusätzlich wird die Transformation der Komponente initialisiert. Dafür wird die Rotation auf [0,0,0], die Translation auf [0,0,0] und die Skalierung auf [1,1,1] gesetzt. Für die Translation wird der Referenztyp auf „absGlobal“ gesetzt. Jede Komponente enthält ein RootSection- und ein RootPositioning-Element welche ebenfalls erstellt werden.

Für die Parameter des RootPositioning-Elements (der Tragflächen-Komponente) werden neben dem Namen, der Beschreibung und der uID die folgenden Werte gewählt:

- **Pfeilung** $sweepAngle = 0^\circ$
- **V-Stellung** $dihedralAngle = 0^\circ$
- **Länge** $length = 1m$
- **Positionierte Sektion** $toSectionUID = rootSectionUID$

Die „toSectionUID“ des RootPositionings wird auf die uID der RootSection gesetzt. Die Rumpf-Komponent wird fast identisch erzeugt, jedoch wird für „positionings“ des Rumpfes eine Pfeilung von $sweepAngle = 90^\circ$ gesetzt.

Die RootSection bekommt ebenfalls Standardwerte für den Namen, die uID und die Beschreibung. Wie auch für die Komponenten wird auch für die RootSection der Transformationsknoten initialisiert. Dabei werden die gleichen Werte verwendet, wie schon für die Komponenten-Transformation (Rotation [0,0,0], Translation [0,0,0], Skalierung [1,1,1]). Da eine „section“ in CPACS mindestens einen „element“-Knoten benötigt wird auch dieser erstellt, ebenfalls mit einem Namen, einer uID, einer Beschreibung und der üblichen Transformation. „element“-Knoten referenzieren immer ein Profil, daher wird für das erstellte „element“ das in den Einstellungen angegebene Standardprofil verwendet (siehe

Abschnitt 3.5.14).

Beim Erstellen neuer Komponenten (Rumpf und Tragflächen) wird auch immer ein entsprechender Part (WingPart oder FuselagePart) erstellt. Die Initialisierung der Parts wird im nächsten Absatz beschrieben.

Neues Part Die Parts sind, wie in Abschnitt 3.3.1 beschrieben, eine Sammlung von CPACS-Knoten. Jedes Part enthält eine „section“ und ein „positioning“, welche analog zu der RootSection und dem RootPositioning einer Komponente initialisiert werden. Allerdings wird für das „positioning“, im Gegensatz zum RootPositioning auch eine „fromSectionUID“ angegeben. Diese wird auf die uID der vorherigen „section“ gesetzt, die beim Erstellen des Parts über das Benutzerinterface angegeben wird, siehe Abschnitt 3.5.5. Zu einem neuen Part gehört aber ebenso ein „segment“-Knoten, der zwei „element“-Knoten zu einem Volumenkörper verbindet. Das „segment“ beinhaltet neben einem Namen, einer uID und einer Beschreibung die Referenzen auf zwei „element“-Knoten. Als „fromElementUID“ wird die uID des ersten „element“-Knotens der vorherigen „section“¹ verwendet.

Neue Elemente Beim Erstellen zusätzlicher „elements“ werden diese analog zu dem „element“-Knoten initialisiert, der beim Erstellen einer „section“ generiert wird. D.h. mit einem Namen, einer uID, einer Beschreibung, einer Transformation und der in den Einstellungen angegebenen Airfoil als Referenz.

Neue Segmente Zusätzliche „segments“ werden ebenfalls wie bereits in den Parts beschrieben erstellt. Ihnen wird ein Name, eine uID, eine Beschreibung sowie zwei Referenzen zu „element“-Knoten zugewiesen. Die „element“-Knoten werden ebenfalls wie im Part initialisiert. Die Referenzen sollten daher nach der Erstellung auf die gewünschten „elements“ eingestellt werden.

3.3.4 Entfernen/Löschen von Komponenten

Dieser Abschnitt behandelt die Vorgänge die beim Löschen von Geometrien ablaufen. Prinzipiell können alle Geometrien, die erstellt werden können, auch wieder gelöscht werden. Jedoch ist es nicht möglich die letzten Elemente in einer Struktur zu löschen, da die Struktur ohne sie keinen Zweck erfüllt. Z.B. kann der letzte WingPart in einer Wing-Komponente nicht gelöscht werden, da eine Tragfläche ohne Geometrie keinen Sinn ergibt. Solange aber noch mehr als ein WingPart in einer Komponente existiert, können beliebige Parts entfernt werden. Dies gilt ebenso für andere Strukturen, deren Sinn von anderen Elementen abhängt. Lediglich Flugzeugentwürfe können immer entfernt

¹ Die vorherige „section“ ist die, die im zugehörigen „positioning“-Knoten als „fromSectionUID“ angegeben ist.

werden.

Dieser Abschnitt beschreibt die Abhängigkeiten die beim Löschen von Geometrien zum Tragen kommen. Die Abhängigkeiten beim Löschen von Geometrien sind sehr einfach zu beschreiben. Elemente, die der zu löschenen Geometrie in der Baumansicht untergeordnet sind werden ebenfalls gelöscht. Das Löschen hat aber keinen Einfluss auf andere Elemente der Baumstruktur.

Nachdem nun die wesentlichen Berechnungen des SGG erläutert wurden, wird in den folgenden Abschnitten auf die Implementierung des Simple Geometry Generators eingegangen.

3.4 Aufbau/Struktur

Der SGG ist komplett in Python geschrieben und kann somit auf jedem Betriebssystem ausgeführt werden, das Python und die benötigten Bibliotheken unterstützt.

Die Implementierung des SGG ist im wesentlichen in drei Teile geteilt. Der erste Teil ist das Layout des grafischen Benutzerinterfaces, über welches der SGG bedient wird. Der zweite Teil ist das vom SGG intern verwendete zentrale Datenmodell. Im dritten Teil wird die Logik des SGG beschrieben. D.h. es werden die wesentlichen Abläufe des SGGs beschrieben, die durch das Interface ausgelöst werden und Operationen auf den Daten des Datenmodells ausführen. Die Elemente der GUI sind mit Funktionen des CPACS-Modells verknüpft, die in der Bibliothek implementiert sind. Im Folgenden werden die drei Teile näher erläutert.

3.4.1 Grafisches Layout

Das grafische Benutzerinterface ist mit Hilfe der PySide-Bibliothek implementiert. PySide ist ein Python-Wrapper für die Qt-Bibliothek. Es ermöglicht es betriebssystemunabhängige grafische Benutzerinterfaces für Python zu erstellen. PySide bietet eine große Bibliothek mit vordefinierten Widgets die miteinander kombiniert werden können um komplexe GUIs zu erstellen.. PySide umfasst grundlegende Widgets wie „QPushButton“, „QComboBox“, „QLineEdit“, „QCheckBox“, „QRadioButton“ und viele mehr. Zum Umfang von PySide gehören aber auch viele speziellere Widgets wie z.B. Widgets für den Dateidialog zum Öffnen oder Speichern von Dateien, Widgets für grafische Ausgaben wie z.B. Plots sowie Layout-Widgets welche die Anordnung von Widgets festlegen. Der gesamte Umfang von PySide kann über die Referenz unter <http://www.pyside.org/docs/pyside/contents.html> eingesehen werden.

Die „QMainWindow“-Klasse bietet eine Vorlage, die als Basis für den SGG verwendet wird. Abbildung 3.12 zeigt die wesentlichen Elemente, die die „QMainWindow“-Klasse beinhaltet. Ein durch „QMainWindow“ erstelltes Interface unterstützt die Verwendung der folgenden Elemente:

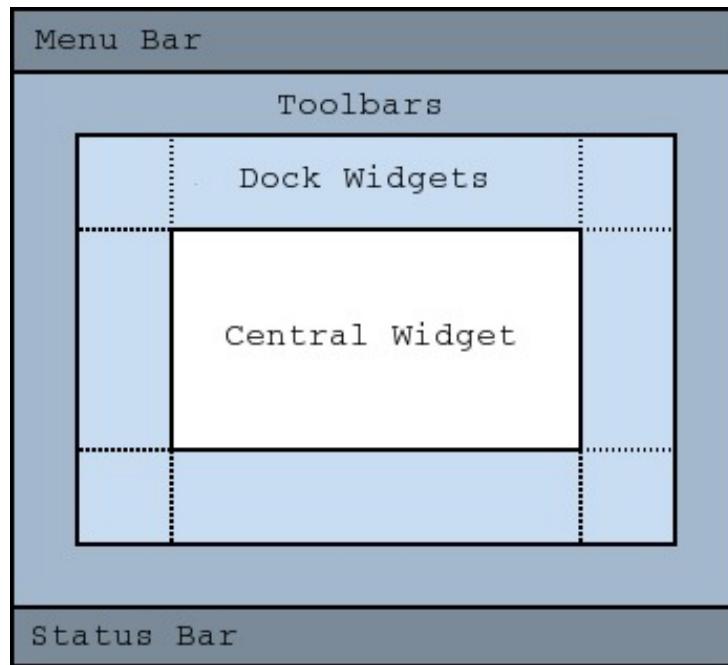


Abbildung 3.12: QMainWindow Layout

Menu Bar „Menu Bar“ ermöglicht eine einfache Implementierung eines üblichen Menüs am Kopf der GUI.

Toolbars Der „Toolbar“-Bereich kann für den schnellen Zugriff auf besondere Funktionen genutzt werden.

Dock-Widgets „Dock-Widgets“ sind Widgets, die sich an verschiedenen Stellen des Interfaces andocken lassen oder frei über/losgelöst vom Hauptfenster schweben können. Die Andockmöglichkeiten der einzelnen Dock-Widgets können bei der Implementierung festgelegt werden.

Central-Widget Das „Central-Widget“ stellt das Widget dar, welches im Focus/Zentrum der Anwendung steht.

Status Bar Der „Status Bar“ kann verwendet werden um Statusmeldungen des Programms auszugeben.

Die SGG-Klasse, welche das SGG-Interface darstellt, ist mittels Vererbung von der „QMainWindow“-Klasse abgeleitet und verwendet das Menü, Dock-Widgets und das Central-Widget.

Die Abbildungen 3.13 bis 3.16 zeigen die Hierarchie der Widgets des SGG. Abbildung 3.13 zeigt, dass der SGG vier Dock-Widgets enthält und das „mainWidget“ als „Central-Widget“. Das „mainWidget“ enthält Instanzen aller Widgets, die je nach Selektion in der Baumansicht („treeView“) angezeigt werden (Abbildungen 3.14 und 3.15). Es dient ausschließlich als Container für die anderen Widgets und ist nur für die Anzeige

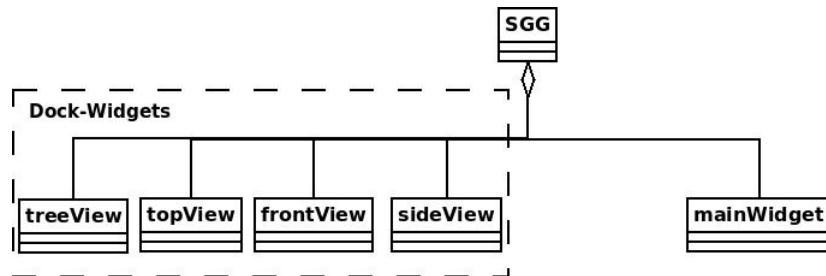


Abbildung 3.13: DockWidgets und mainWidget des SGG

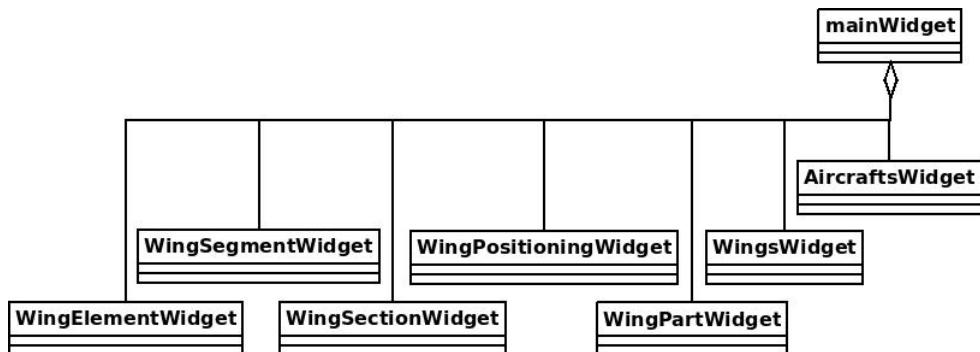


Abbildung 3.14: Widgets der Wing-Komponente des SGG

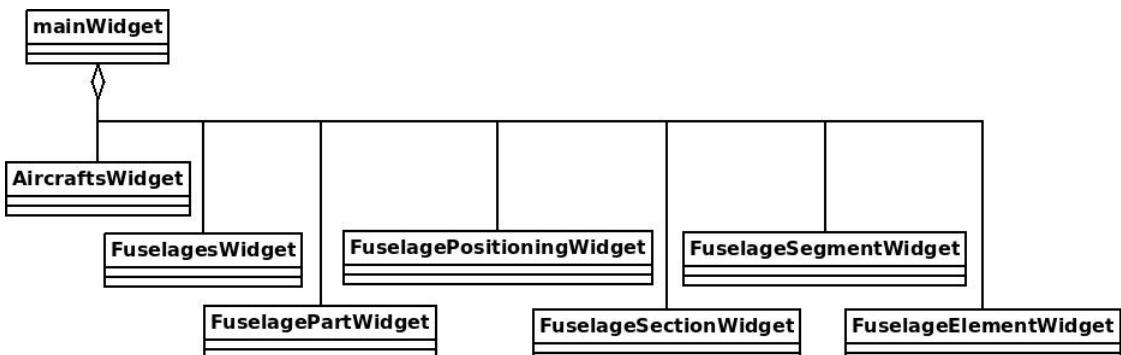


Abbildung 3.15: Widgets der Fuselage-Komponente des SGG

des jeweils ausgewählten Widgets zuständig. Für diese Aufgabe ist in PySide die Layoutklasse „QStackedLayout“ definiert, welche für das „mainWidget“ verwendet wird. Die in Abbildung 3.16 gezeigte „EditTransformation“-Klasse wird in Widgets eingebunden, die einen „transformation“-Knoten beinhalten.

Bisher wurde das allgemeine Layout des SGG beschrieben. Um das entwickelte Layout zu verwenden, ist es notwendig die grafischen Elemente mit implementierten Funktionen zu verknüpfen. Zu diesem Zweck steht in PySide ein Signal&Slot-System zur Verfügung. Das Signal&Slot System ist eine Art Ereignissystem, das es ermöglicht Funktionen durch Ereignisse auszuführen, die z.B. vom Benutzer über das Benutzeroberfläche ausgelöst werden. Signale in PySide sind Objekte (Instanzen einer Klasse) mit denen sich Funktionen

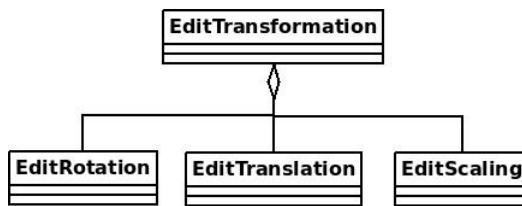


Abbildung 3.16: EditTransformationWidget des SGG

verbinden können. Wird das Signal ausgelöst, so werden alle mit diesem Signal verbundenen Funktionen automatisch aufgerufen. Ein Objekt der „QPushButton“-Klasse hat z.B. ein vordefiniertes „clicked“-Signal an welches man Funktionen binden kann. Diese werden dann beim Drücken des Knopfes mit dem Cursor (im Benutzerinterface) aufgerufen. Funktionen die Signale empfangen werden in PySide mittels Dekorator¹ gekennzeichnet. Signale übermitteln in der Regel Parameter an die von ihnen aufgerufenen Slots. Der Typ des Parameters wird durch den Dekorator angegeben.

Das mittels PySide implementierte grafische Benutzerinterface beinhaltet im wesentlichen vier Teile, die in Abbildung 3.17 dargestellt sind.

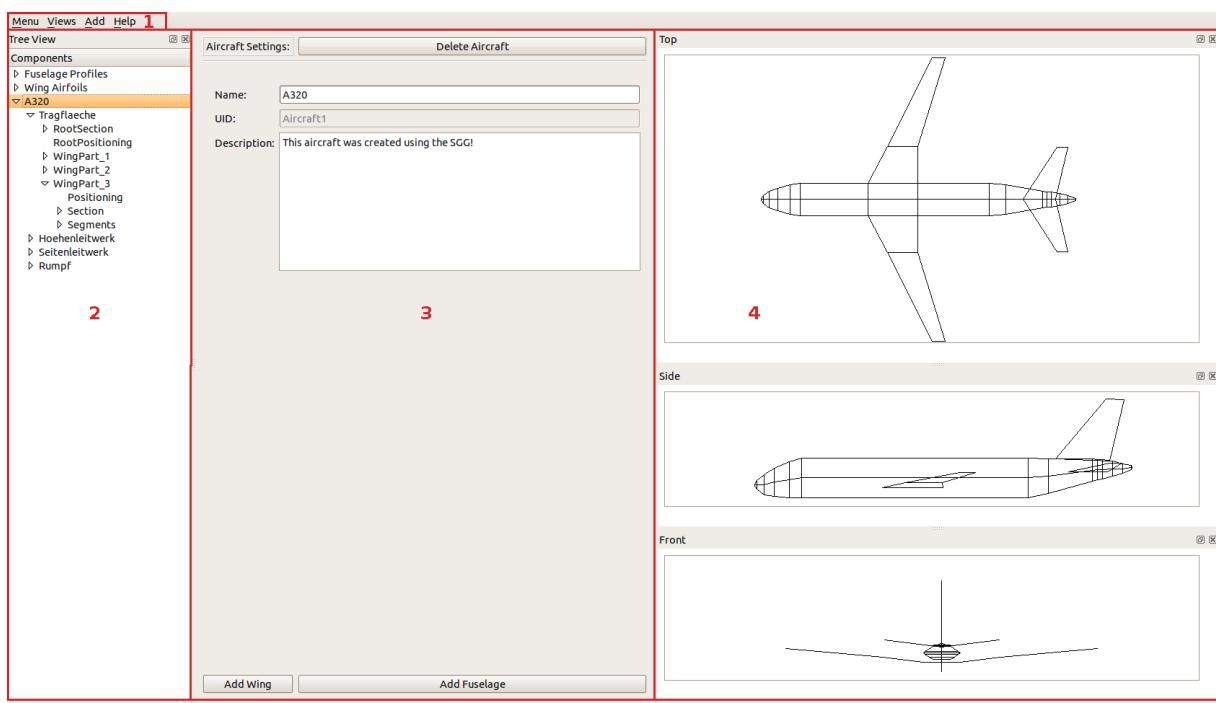


Abbildung 3.17: Hauptansicht des SGG

In dem Hauptfenster des SGG, welches aus einem betriebssystems-typischen Rahmen

¹ Das Verständnis von Dekoratoren für Python setzt tiefere Programmierkenntnisse voraus und ist für das weitere Verständnis dieser Arbeit nicht von Bedeutung. Für eine Beschreibung von Dekoratoren wird an dieser Stelle auf [Kai08] verwiesen.

besteht, befindet sich ein Hauptmenu (Nr. 1) mit Funktionen wie dem Laden und Speichern von Dateien, dem Ein- und Ausblenden von Ansichten sowie dem Aufrufen des Einstellungsfensters.

Am linken Rand ist die Baumansicht der Geometriestruktur (Nr. 2) zu sehen, welches eine einfache und intuitive Navigation durch die Geometriestruktur und die Auswahl einzelner Elemente zum Bearbeiten ermöglicht.

In der Mitte des Fensters befindet sich das Hauptwidget (Nr. 3). In diesem Widget werden, je nach Selektion in der Baumansicht, die Widgets zum Bearbeiten der Geometrien angezeigt.

Rechts von dem Hauptwidget sind die drei Ansichten (Nr. 4) angeordnet, die eine vereinfachte Darstellung der Flugzeuggeometrien darstellen. Bei mehreren Flugzeugentwürfen zeigen die Ansichten immer die Flugzeuggeometrie, dessen Flugzeugknoten in der Baumansicht zuletzt ausgewählt wurde.

Eine detaillierte Beschreibung des Menüs und der Widgets befindet sich in Abschnitt 3.5. Bei der Implementierung eines grafischen Benutzerinterfaces, welches Zugriff auf eine große Datenstruktur bietet, ist es oft nicht zu vermeiden, dass verschiedene Widgets auf die selben Daten zugreifen. Bei rein lesenden Zugriffen, bei denen keine Daten verändert werden, stellt dies keine Schwierigkeit dar. Wenn jedoch Widgets Änderungen an Daten vornehmen die von anderen Widgets ebenfalls verwendet werden, ist es wichtig die Konsistenz der Daten zu bewahren. Daher ist es notwendig die Widgets immer mit aktuellen Daten zu versorgen. Zu diesem Zweck wird für den SGG das im folgenden Abschnitt beschriebene zentrale Datenmodell verwendet.

3.4.2 Datenmodell

Um sicherzustellen, dass die verschiedenen Widgets immer auf die aktuellen Daten zugreifen, wird innerhalb des SGG ein zentrales Datenmodell verwendet. Die in Abbildung 3.18 gezeigte cpacsModel-Klasse stellt dieses zentrale Datenmodell dar. Die

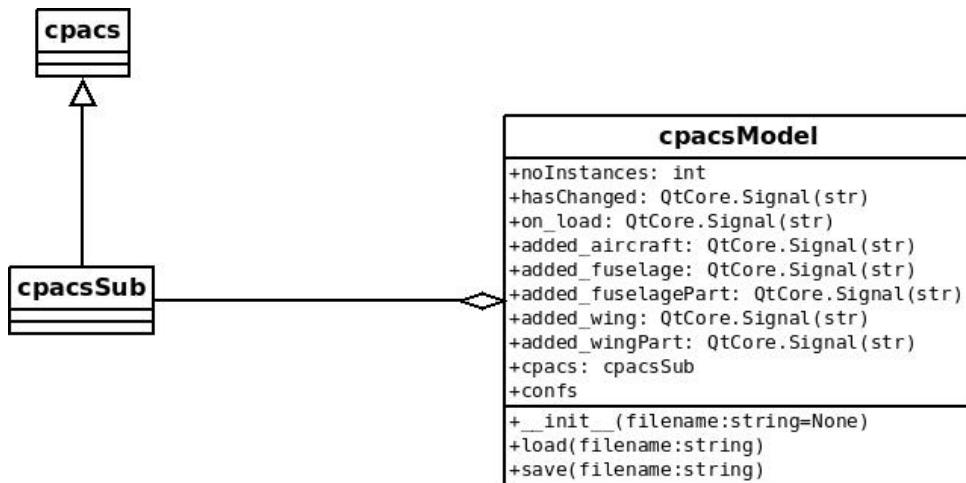


Abbildung 3.18: cpacsModel Klasse

cpacsModel-Klasse beinhaltet ein cpacs-Attribut, ein Objekt der cpacsSub-Klasse welche die cpacs-Struktur beinhaltet, ein config-Attribut mit Konfigurationsdaten für den SGG, ein viewData-Attribut mit den Koordinaten für die Ansichten, sowie verschiedene PySide-Signale (siehe Abschnitt 3.4.1).

3.4.2.1 cpacs-Attribut

Das cpacs-Attribut ist ein Objekt der „cpacsSub“-Klasse, welche mittels generateDS, einem Programm zum automatischen Erzeugen von Klassendefinitionen (siehe Anhang A.1), aus der CPACS-Schemadefinition generiert wurde. Die mittels generateDS erzeugten Klassen beinhalten die in der Programmierung üblichen Set- und Get-Methoden sowie Funktionen zum Import und Export aus/in XML-Dateien.

Die erstellten Klassen können um zusätzliche Funktionalitäten erweitert werden indem neue Methoden in die Klassendefinitionen eingefügt werden. In der Dokumentation von generateDS wird empfohlen zur Erweiterung der Klassen „subclasses“ zu Verwenden, welche von den generierten Klassen abgeleitete Klassen sind [Kuh12]. Eine solche Datei mit den „subclasses“ kann ebenfalls automatisch von generateDS erzeugt werden. Die Verwendung der „subclasses“ hat die folgenden Vorteile:

- Es trennt den Programmcode von der CPACS-Definition, was zu übersichtlicherem Code und besserer Modularität/Austauschbarkeit führt.
- Da die Datei mit den erstellten Superklassen nicht verändert wird kann diese wiederverwendet oder mit einer aktuelleren Version ausgetauscht werden. Der Austausch der generierten Superklassen mit einer neuen CPACS-Version kann auch Änderungen in den „subclasses“ notwendig machen.

Für den SGG wurde daher die Verwendung einer „cpacsSub“-Datei beschlossen, welche die abgeleiteten Klassen (mit dem Suffix „Sub“) beinhaltet. Eine solche Datei kann durch das Hinzufügen von Funktionalitäten auch als CPACS-Bibliothek für die Implementierung weiterer CPACS-Programme in Python verwendet werden. Um die Wiederverwendbarkeit von Code zu erleichtern, werden alle Kernfunktionen, die nicht direkt mit der grafischen Benutzerschnittstelle zu tun haben in dieser Bibliothek implementiert. Die Zuordnung von Funktionen zu den jeweiligen Klassen unterstreicht die Objektorientierung beim Aufbau der Bibliothek. Funktionen zum Erstellen und Initialisieren von CPACS-Knoten sind in den jeweiligen Klassen implementiert und vereinfachen den Umgang mit der CPACS-Struktur. generateDS ermöglicht es mittels der generierten Funktionen „parse“ und „write“ auf XML-Dateien zuzugreifen, die der Schemadefinition, welche zur Erstellung der Klassen verwendet wurde, genügen. Das Wurzelement der CPACS-Struktur ist das „cpacs“-Element, welches in der abgeleiteten Variante als „cpacsSub“-Klasse zur Verfügung steht. Durch die Verwendung dieser „cpacsSub“-Klasse können alle Änderungen an dem CPACS-Datensatz direkt in das „cpacsSub“-Objekt überführt werden. Die Elemente der grafischen Benutzerschnittstelle verwenden das cpacs-Objekt um immer auf die aktuellen Daten

zuzugreifen.

Die Verbindung zum TIGLViewer ist ebenfalls über das cpacs-Attribut realisiert. Über die Exportfunktionalität wird jeweils das aktuell betrachtete Modell und die Profile in ein CPACS-Datei exportiert, die vom TIGLViewer überwacht wird. Bei Änderungen an der Datei aktualisiert sich der TIGLViewer automatisch. Näheres über die Verbindung zum TIGLViewer befindet sich in Abschnitt 3.6.

3.4.2.2 config-Attribut

Das config-Attribut enthält Konfigurationsdaten für den SGG. Es ist in Python als „Dictionary“ implementiert und enthält die folgenden Elemente:

- TIGLViewerPath
- CPACSLibPath
- Airfoil
- Profile
- top
 - uselImage
 - imagePath
 - width
 - scale
- side
 - uselImage
 - imagePath
 - width
 - scale
- front
 - uselImage
 - imagePath
 - width
 - scale

Der „TIGLViewerPath“ gibt den Pfad zur lokalen Installation des TIGLViewers an. Wenn der TIGLViewer über die zuständige Menüfunktion nicht startet, sollte dieser Wert überprüft werden.

Der „CPACSLibPath“ gibt den Pfad zur vom SGG verwendeten CPACS-Bibliothek an. Dies bietet die Möglichkeit für verschiedene Entwurfsaufgaben angepasste Bibliotheken zu verwenden.

In „Airfoil“ wird die uID des vom SGG standardmäßig verwendeten Flügelprofil angegeben. Profile dient der Angabe eines Standardprofils für Rumpfquerschnitte.

Die drei Elemente top, side und front sind selber „Dictionarys“, die Informationen zu den Ansichten enthalten. Das „useImage“-Attribut enthält einen boolschen Wert der angibt, ob für die jeweilige Ansicht ein Hintergrundbild verwendet wird. Der Pfad zu dem jeweiligen Bild ist in „imagePath“ abgelegt. Über „width“ kann festgelegt werden, wie Breit das Bild im SGG angezeigt wird. Um dem SGG mitzuteilen in welchem Maßstab das Bild abgelegt ist, wird „scale“ verwendet. „scale“ wird in Pixel pro Meter angegeben, d.h. im Bild werden für einen Meter des abgebildeten Flugzeuges die angegebene Menge an Pixeln verwendet. Dieses Attribut dient der automatischen Skalierung der gezeichneten Geometrie auf das Hintergrundbild.

3.4.2.3 **viewData**

Das `viewData`-Attribut enthält eine Koordinatenliste der Geometrie des Flugzeugentwurfs. Die Liste besteht aus Punktepaaren deren Verbindungslien in den Ansichten dargestellt werden. Sie dient somit als Datensammlung für die drei Ansichten. Da es sich bei den Punkten um dreidimensionale Punkte handelt können die gleichen Punkte für alle drei Ansichten verwendet werden. Je nach Ansicht wird jeweils eine Komponente ignoriert um die Projektionen der Punkte auf die Ebenen darzustellen. Bei Änderungen der Geometrie des Entwurfs werden diese Daten neu berechnet. Die Berechnung der Koordinaten wird in Abschnitt 3.3.1 beschrieben.

3.4.2.4 **PySide-Signale**

PySide-Signale werden verwendet um die Kommunikation zwischen Widgets zu ermöglichen. Anhand von Signalen können Widgets über Ereignisse in anderen Widgets informiert werden. Im SGG werden die Signale verwendet um Widgets über Änderungen am Datenmodell zu informieren, die von anderen Widgets eingeleitet wurden. Die im cpacs-Model implementierten Signale werden emittiert wenn die in folgenden angegebenen Ereignisse eintreten.

`changed_geometry` Dieses Signal wird verwendet um Änderungen der Geometrie zu erkennen. Es wird nur von der `cpacsModel`-Klasse selbst empfangen um die Koordinatenberechnung neu durchzuführen. Gesendet wird es von den „`change_...`“-Methoden der Widget-Klassen. D.h. bei Änderungen über das Benutzerinterface, die die Geometrie betreffen (nicht dazu gehört z.B. die Änderung von Namen oder Beschreibungen).

changed_viewData Dieses Signal dient der Aktualisierung der Ansichten. Es wird vom cpacsModel am Ende der calc_viewData()-Methode aufgerufen um die Ansichten über Änderungen der Koordinaten zu informieren.

changed_config Dieses Signal wird verwendet um die Ansichten über Änderungen in den Einstellungen zu informieren. Das Signal wird beim Laden von Konfigurationsdateien (cpacsModel.loadDefaultConfig(), cpacsModel.loadConfig()) und beim Ändern von Einstellungen (configWidget.setConfig()) gesendet. Empfangen wird es von den Ansichten (view2D.reset()).

show_widget Dieses Signal wird dazu verwendet Widgets von beliebigen anderen Widgets aus anzuzeigen. Das ermöglicht es nach dem Löschen von Komponenten die jeweiligen übergeordneten Widgets anzuzeigen. Diese Funktion wird in den delete-Methoden der Widgets gesendet und vom mainWidget (mainWidget.show_editWidget()) empfangen.

delete_aircraft Dieses Signal sorgt für das vollständige Entfernen eines Flugzeuges im SGG. Da im TreeView ein Verweis auf jedes bestehende Flugzeug abgelegt ist, wird das Signal dort empfangen und der Verweis gelöscht.

Beim Starten des SGG wird eine Instanz der cpacsModel-Klasse erzeugt und jedes Widget bekommt eine Referenz auf diese Instanz übergeben. Auf diese Weise werden Änderungen, die durch ein Widget am Datensatz durchgeführt werden für alle Widgets sichtbar. Durch die jeweiligen Signale werden Widgets gezielt über Änderungen informiert, die für diese relevant sind.

3.4.3 Programmlogik

Die Programmlogik beschreibt was hinter den Kulissen der grafischen Benutzeroberfläche abläuft. Sie besteht aus einem komplexen Zusammenspiel von Funktionen. Dieser Abschnitt liefert eine Übersicht über die wesentlichen Abläufe im SGG.

Der SGG hat eine Hauptroutine (Standard PySide Application) die das grafische Interface aus Abschnitt 3.5 steuert. Beim Starten des SGG wird diese Hauptroutine aufgerufen, der SGG wird initialisiert und wartet anschließend auf Benutzerinteraktionen. Während der Initialisierung wird bereits eine Instanz jedes zum SGG gehörenden Widgets erzeugt und auf einem Stapel abgelegt. Wann immer ein Widget angezeigt werden soll, wird es in den Vordergrund geholt. Dabei wird der CPACS-Pfad angegeben, der dem Widget mitteilt, an welcher Stelle des zentralen Datenmodells die zugehörigen Daten liegen. Jedes mal wenn ein Widget neu angezeigt wird, also beim Wechsel zwischen Widgets, werden aktuelle Werte aus dem cpacsModel verwendet.

Dieses Vorgehen sorgt dafür, dass die Daten der EditWidgets¹ immer konsistent sind.

¹ Die Widgets zum Bearbeiten von CPACS-Daten, siehe Abbildung 3.17 (Nr. 3).

Allerdings ist dies nur bei Widgets möglich, die aus dem Hintergrund heraus wieder angezeigt werden. Widgets, wie die Baumansicht oder die Seitenansichten, die parallel zu den EditWidgets angezeigt werden, müssen auf andere Weise über Änderungen informiert werden. Zu diesem Zweck wird im SGG das in Abschnitt 3.4.1 vorgestellte Signal&Slot-System verwendet, welches von PySide bereitgestellt wird. Die im SGG verwendeten Signale wurden bereits im letzten Abschnitt beschrieben.

Die Bedienelemente der Widgets sind in der Regel direkt mit CPACS-Attributen des zentralen Datenmodells verbunden. Parameter die nicht direkt in CPACS abgelegt sind, wie z.B. die Spannweite oder die Flügelfläche, werden bei der Eingabe, je nach gewähltem Parametersatz, in CPACS-Parameter umgerechnet.

Der Wechsel zwischen den Widgets geschieht über die Baumansicht am linken Rand. Die dort gezeigten Elemente bieten eine Übersicht über die in Parts gegliederte CPACS-Struktur. Diese Struktur ist eine für den SGG verwendete Möglichkeit CPACS-Geometriedaten darzustellen.

Durch Verwenden des Programm-Menüs werden direkt Aktionen ausgeführt wie z.B. das Laden und Speichern von Dateien, das Aufrufen des Einstellungsfensters oder das Erstellen neuer Flugzeugentwürfe.

Abbildung 3.19 stellt die beschriebene Kommunikation zwischen den Widgets dar. Sie

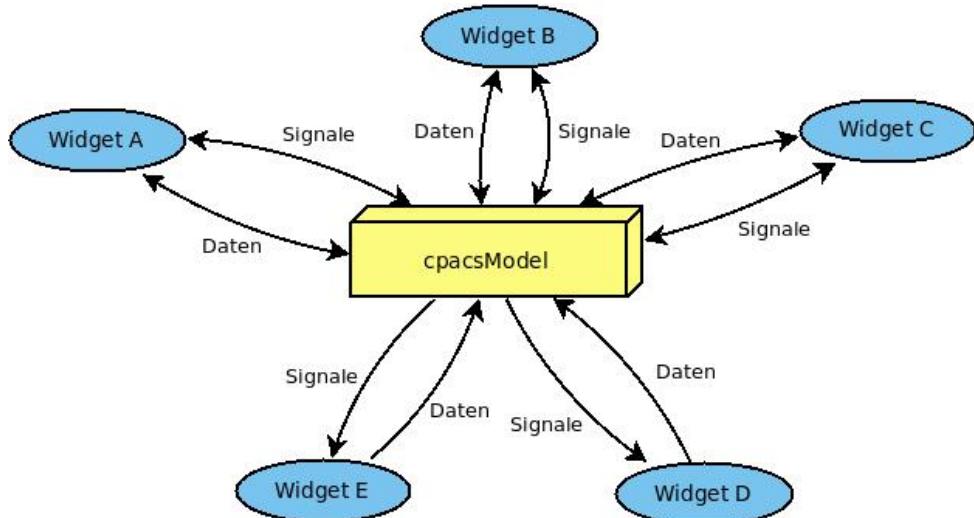


Abbildung 3.19: Kommunikation der Widgets

veranschaulicht, dass jedes Widget Zugriff auf das zentrale cpacsModel hat und auf die dort liegenden Daten lesend und schreibend zugreifen kann. Die zweite Verbindung zum cpacsModel stellt den Signalaustausch dar, über den Widgets Ereignisse auslösen und empfangen können um miteinander zu kommunizieren.

3.5 Grafische Benutzerschnittstelle

Die grafische Benutzerschnittstelle dient der Interaktion des Benutzers mit dem SGG. In diesem Abschnitt werden die einzelnen Widgets des SGG detailliert beschrieben. Die

Widgets zum Bearbeiten der Rumpf- und Tragflächengeometrien sind fast identisch. Die Implementierungen der jeweiligen Klassen sind daher von einer gemeinsamen Superklasse abgeleitet. In diesem Abschnitt werden daher die Superklassen beschrieben und die Unterschiede zwischen Rumpf- und Tragflächenimplementierung hervorgehoben. Für die Abbildungen werden die „Wing“-Versionen der Widgets verwendet.

Da sich das WingPartWidget wegen der unterschiedlichen Parameter stark vom FuselagePartWidget unterscheidet werden diese beiden Widgets jedoch separat aufgeführt.

3.5.1 Menü

Abbildung 3.20 zeigt die Struktur des Hauptmenüs des SGG. Tabelle 3.5 beschreibt, was

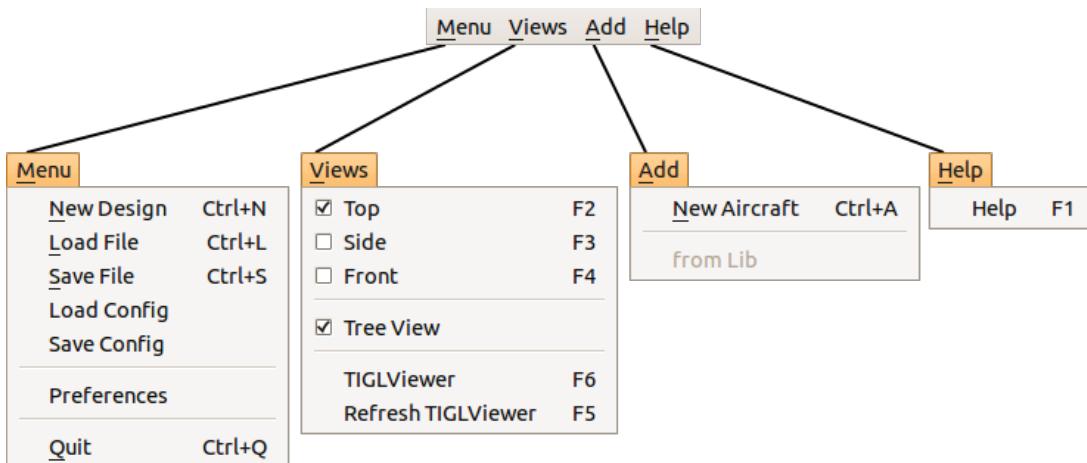


Abbildung 3.20: Hauptmenü

sich hinter den einzelnen Menüeinträgen verbirgt.

3.5.2 Baumansicht

Am linken Rand des SGG ist eine Baumansicht der Geometrien, wie in Abbildung 3.21 dargestellt, zu sehen. Die Baumansicht zeigt eine hierarchische Struktur der Geometrien und dient der Selektion einzelner Geometrien. Das Widget zum Bearbeiten der Geometrien wird je nach Auswahl in der Baumansicht angepasst.

Die Baumansicht basiert auf einem Modell, dass die angezeigte Struktur darstellt. Um die Baumstruktur umzusetzen, wurde die in den Abbildungen 3.22 bis 3.24 dargestellte Klassenstruktur implementiert. Die einzelnen Items sind von der in Abbildung 3.25 dargestellten Klasse abgeleitet, und enthält die CPACS-Pfade zu den Knoten, die für das jeweilige Widget benötigt werden.

3.5.3 TransformationWidget

Das TransformationWidget aus Abbildung 3.26 dient der Bearbeitung der Translation, Rotation und Skalierung von Geometrien. Es wird von verschiedenen Widgets, wie

File	Beinhaltet Funktionen für den Umgang mit Dateien und einen Eintrag zum Beenden des Programms.
New Design	Entfernt alle erstellten Geometrien und stellt den Ausgangszustand des SGG wieder her.
Load	Ruft den Dateidialog zum Laden einer CPACS-Datei auf.
Save	Ruft den Dateidialog zum Speichern der aktuellen CPACS-Struktur in eine Datei auf.
Load Config	Ruft den Dateidialog zum Laden einer Konfigurationsdatei auf.
Save Config	Ruft den Dateidialog zum Speichern der aktuellen Konfigurationen in eine Datei auf.
Preferences	Ruft das Fenster zur Konfiguration der Einstellungen auf.
Quit	Beendet das Programm.
Views	Dient dem Anzeigen/Ausblenden von Ansichten.
Front	Zeigt die Vorderansicht, oder blendet sie aus.
Side	Zeigt die Seitenansicht, oder blendet sie aus.
Top	Zeigt die Draufsicht (von oben), oder blendet sie aus.
TreeView	Zeigt die Baumstruktur oder blendet sie aus.
TIGLViewer	Startet den TIGLViewer.
TIGLViewer	Aktualisiert die Geometrien im TIGLViewer.
Add	In dieser Menüleiste stehen Funktionen zur Verfügung, die neue Komponenten zur CPACS-Struktur hinzufügen.
Aircraft	Fügt einen leeren Flugzeugentwurf der aktuellen CPACS-Struktur hinzu.
Help	Diese Menüleiste enthält Hilfe zum SGG
About	Zeigt ein Fenster mit allgemeinen Informationen zum SGG an.

Tabelle 3.5: Menüeinträge

z.B. dem WingWidget verwendet. Die Werte können entweder über die „SpinBox“-Elemente oder die „Slider“ eingestellt werden. Die Grenzen der Werte können über die Einstellungen im Menü **Menu->Preferences** angepasst werden.

3.5.4 AircraftWidget

Das in Abbildung 3.27 dargestellte AircraftWidget stellt die CPACS-Elemente des „model“-Knotens zum Bearbeiten bereit. Dazu gehören der Name, die Beschreibung und die uID des Flugzeugentwurfs. Außerdem können in diesem Widget neue Rumpf- und Tragflächen- bzw. Leitwerkskomponenten durch Drücken der entsprechenden PushButtons (Add Fuselage, Add Wing) erstellt werden. Die Erzeugung neuer Komponenten ist in Abschnitt 3.3.3 beschrieben. Das AircraftWidget wird angezeigt, wenn in der Baumanansicht ein Flugzeugentwurf ausgewählt ist. Durch Drücken auf „Delete Aircraft“ wird der komplette Flugzeugentwurf unwiderruflich gelöscht.

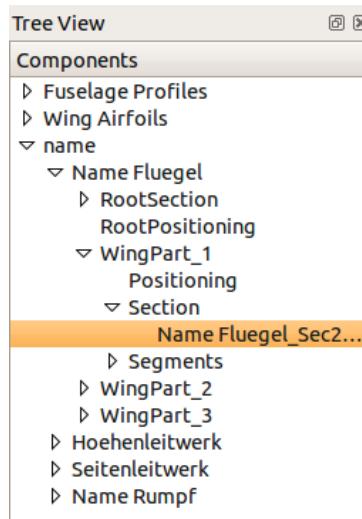


Abbildung 3.21: Baumansicht

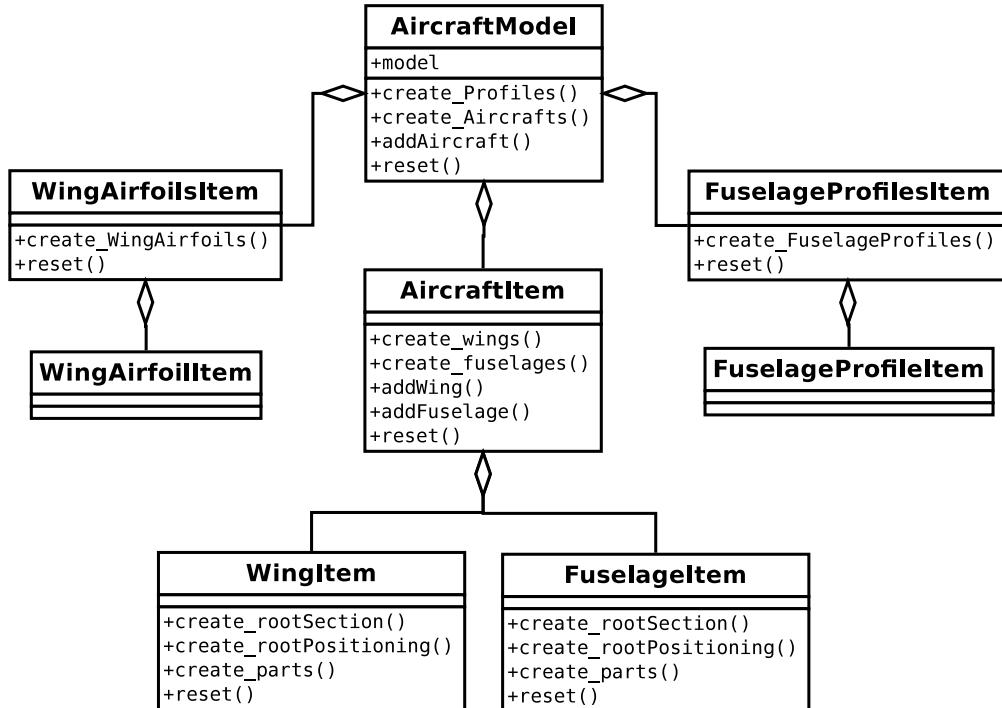


Abbildung 3.22: AircraftModel Struktur

3.5.5 ComponentWidget

Das ComponentWidget, welches in Abbildung 3.28 dargestellt ist, ermöglicht den Zugriff auf die Attribute einer Komponente. Die Attribute der Komponente sind neben dem Namen, der Beschreibung und der uID, auch die parentUID. Über die parentUID kann eine andere Komponente gewählt werden bezüglich derer die aktuelle Komponente platziert wird.

Des Weiteren beinhaltet dieses Widget eine Instanz des TransformationWidgets um die

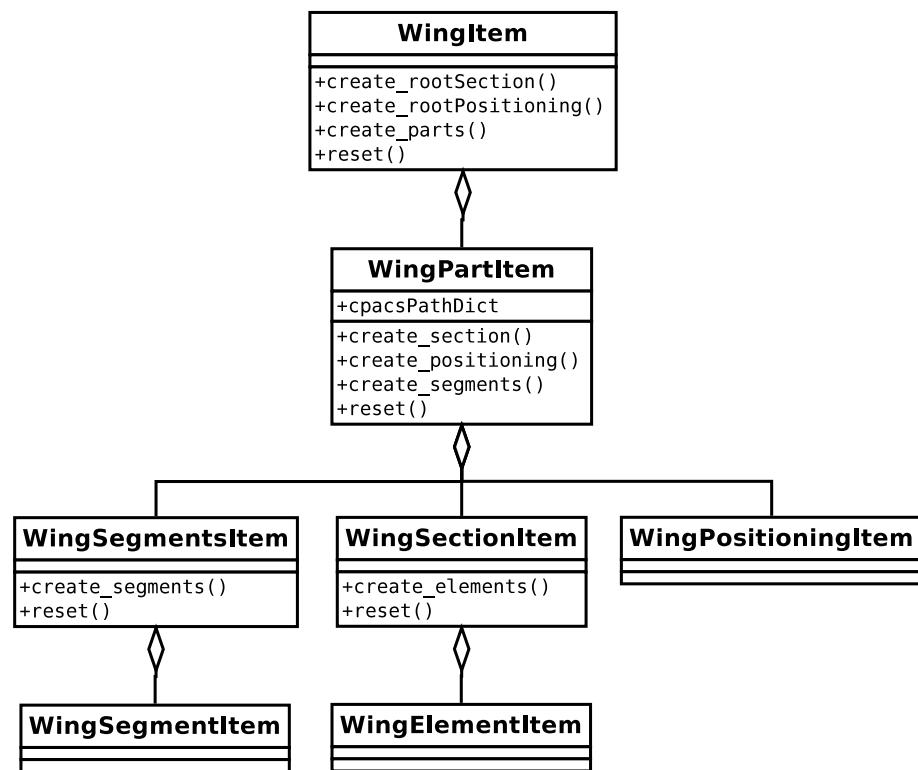


Abbildung 3.23: WingItem Struktur

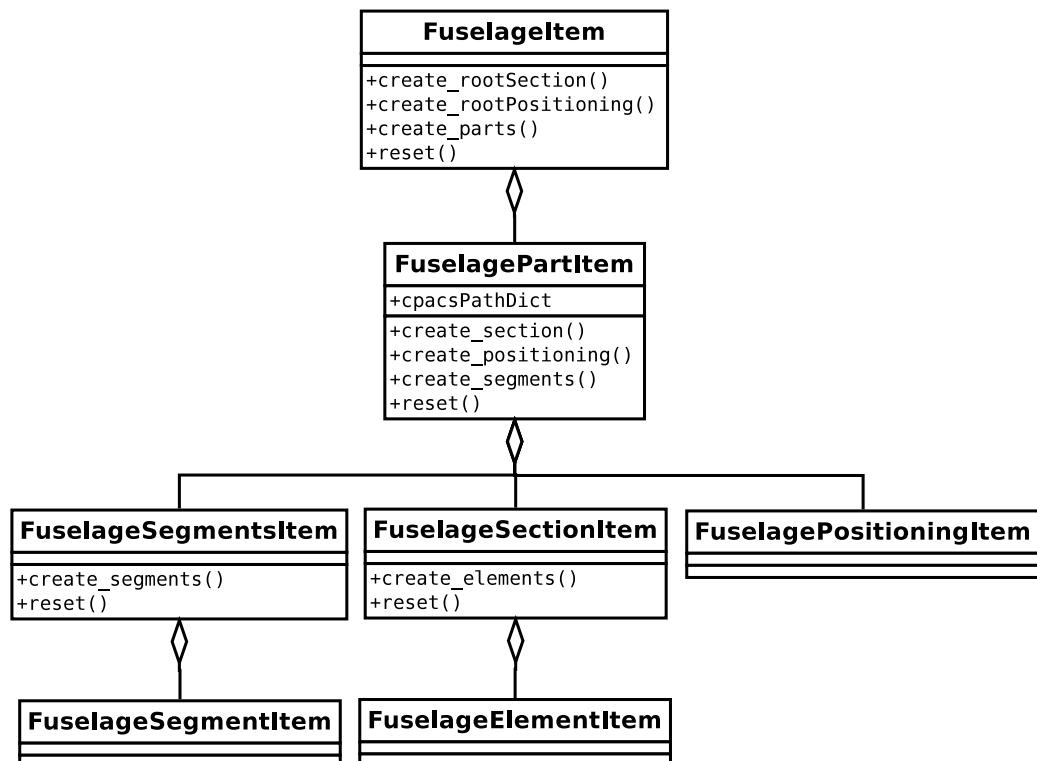


Abbildung 3.24: FuselageItem Struktur

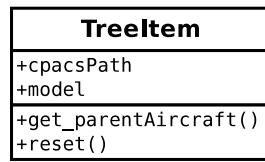


Abbildung 3.25: TreeItem Klasse

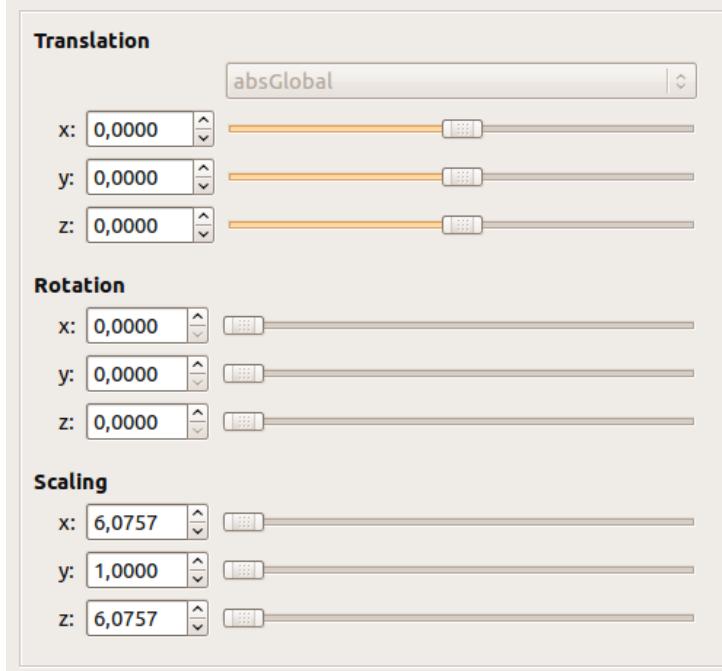


Abbildung 3.26: TransformationWidget

Transformationen („Translation“, „Orientation“, „Scaling“) der Komponente zu bearbeiten. Der mit „insert Part after“ bezeichnete PushButton ermöglicht es der Komponente neue Parts der Komponente hinzuzufügen. Die ComboBox rechts des Knopfes gibt an, an welches „section“-Element der neue Part angehängt wird. Der Part wird mit Standardparametern erstellt und kann über den TreeView ausgewählt und bearbeitet werden. Die Komponente wird durch Drücken auf „Delete Wing“ für das WingWidget und „Delete Fuselage“ für das FuselageWidget vom Flugzeugentwurf entfernt.

3.5.6 WingPartWidget

Das WingPartWidget bietet eine vereinfachte Möglichkeit auf Flügelsegmente zuzugreifen. Über typische Entwurfsparameter können so einzelne Flügelsegmente konfiguriert werden. In Abbildung 3.29 wird das WingPartWidget gezeigt. Die dort aufgeführten Parameter Sweep (Pfeilung), Dihedral (V-Stellung), Twist (Profilverwindung), Span (Spannweite), Taper Ratio (Zuspitzung), Sref (Referenzfläche) und Aspect Ratio (Streckung) können je nach Einstellung in unterschiedlichen Kombinationen verwendet werden. Die Parameterkombinationen sind in Tabelle 3.3 abgebildet. Über die unter „Parameters“ ste-

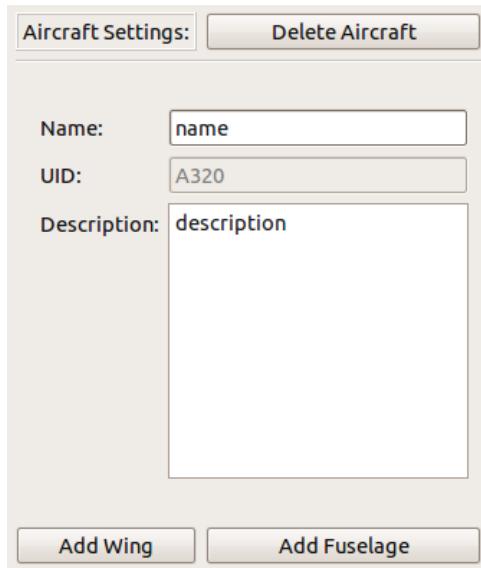


Abbildung 3.27: AircraftWidget

hende ComboBox kann zwischen den verschiedenen Parameterkombinationen gewählt werden. Nicht verwendete Parameter werden deaktiviert, jedoch bei Änderungen der übrigen Parameter erneut berechnet, sodass der aktuelle Wert angezeigt wird. Mittels der „Airfoil“ ComboBox kann das für den Querschnitt des WingParts verwendete Profil ausgewählt werden. Das Profil des vorherigen Querschnitts bleibt dabei unverändert. Ein WingPart enthält genau eine „section“ welche jedoch mehrere „element“-Knoten beinhalten kann. Daher wird zur Berechnung der Parameter immer das toElement des unter „Main Segment“ angezeigten „segment“-Knotens verwendet.

Die dargestellten Parameter werden aus den CPACS-Parametern wie folgt berechnet:

- Die Pfeilung, wird wie in CPACS in der Tragflächenebene in Grad angegeben. Das ist der Winkel zur um die V-Stellung rotierten Y-Achse.
- Die V-Stellung wird wie in CPACS in der globalen Tragflächen-Y-Achse in Grad angegeben.
- Die geometrische Profilverwindung wird als Winkel zwischen dem inneren und äußeren Querschnitt angegeben. Sie stellt also die Verwindung des jeweiligen Parts dar und nicht den globalen Anstellwinkel.
- Die Spannweite gibt die Länge des Parts in globaler Tragflächen-Y-Achse dar. Für symmetrische Tragflächen ist die Spannweite die Summe des originalen und des gespiegelten Parts.
- Die Zuspitzung wird aus den Profiltiefen zweier Elemente berechnet. Als Elemente werden die „element“-Knoten verwendet, die im ersten „segment“-Knoten an-

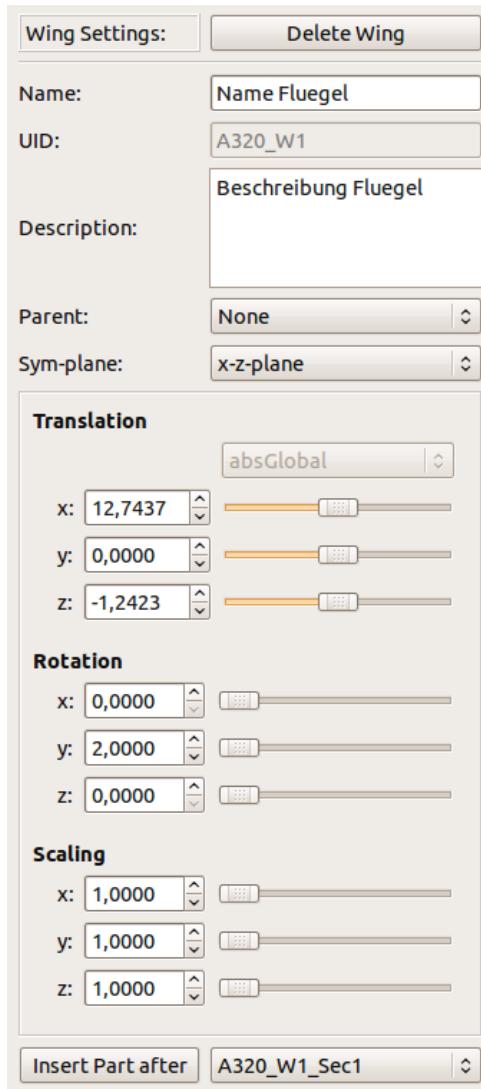


Abbildung 3.28: WingWidget

gegeben werden. Zur Berechnung der Profiltiefen werden die Skalierungen der „element“- und „section“-Knoten berücksichtigt.

- Als Referenz-Flügelfläche wird die, durch die beiden Querschnitte aufgespannte Trapezfläche verwendet.
- Die Streckung wird per Definition als $\Lambda = \frac{\text{span}^2}{S_{ref}}$ berechnet.

Eine detailliertere Beschreibung der Parameter und deren Berechnung befindet sich in Abschnitt 3.3.1.1.

3.5.7 FuselagePartWidget

Das FuselagePartWidget bietet in Analogie zum WingPartWidget Zugriff auf die Parameter des FuselageParts. Die Parameter ermöglichen es die Geometrie des FuselageParts

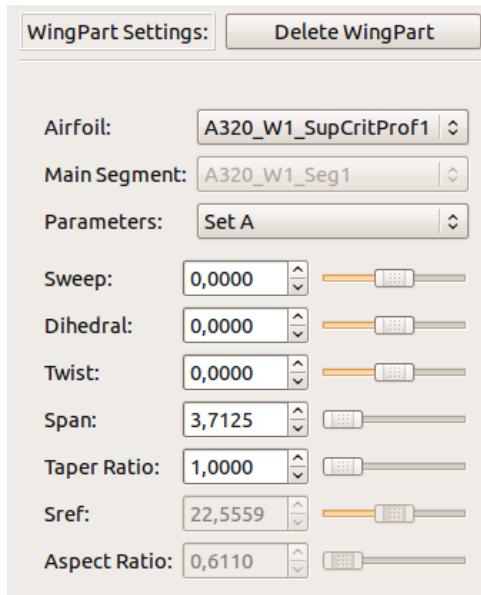


Abbildung 3.29: WingPartWidget

anhand von intuitiven Parametern zu definieren. Abbildung 3.30 zeigt das FuselagePart-Widget mit den im FuselagePart definierten Parametern. Auch für den FuselagePart

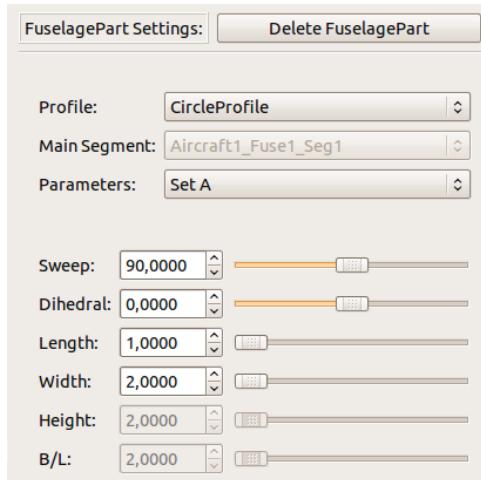


Abbildung 3.30: FuselagePartWidget

stehen unterschiedliche Parameterkombinationen zur Verfügung zwischen denen unter „Parameters“ gewählt werden kann. Wie schon für das WingWidget werden in einer Parameterkombination nicht verwendete Parameter für die Bearbeitung deaktiviert, deren Wert aber aktualisiert. Das Querschnittsprofil für den FuselagePart wird über die ComboBox unter „Profile“ ausgewählt.

Die Parameter der FuselageParts haben die folgende Bedeutung:

- Die Pfeilung („Sweep“), wird wie in CPACS als Winkel zur um die V-Stellung rotierten Y-Achse in Grad angegeben.

- Die V-Stellung („Dihedral“) wird wie in CPACS als Winkel zur globalen Y-Achse in Grad angegeben.
- Unter „Length“ wird die Länge des FuselageParts angegeben. Es handelt sich dabei um die Länge des durch die Pfeilung und V-Stellung aufgespannten Vektors.
- „Width“ gibt die Breite des Rumpfquerschnittes an.
- Über „Height“ kann die Höhe des Querschnitts angegeben werden.
- B/L ist das Verhältnis der Querschnittsbreite zur Länge des FuselageParts.

Details zu den FuselagePart-Parametern befinden sich in Abschnitt 3.3.1.2

3.5.8 SectionWidget

Das SectionWidget, dargestellt in Abbildung 3.31, ermöglicht den direkten Zugriff auf eine CPACS-Section. In diesem Widget können der Name, die uID, die Beschreibung und die Transformationen der „section“ bearbeitet werden. Das SectionWidget enthält außerdem eine Instanz des TransformationWidgets, welche den Zugriff auf die Translation, Rotation und Skalierung der „section“ ermöglichen. Über den „Add Element“-Knopf können neue „element“-Knoten innerhalb der „section“ erstellt werden. Dadurch entstehen „Multi-Element-Sections“, die es ermöglichen eine Struktur aufzuteilen um z.B. Multi-Winglets [Ber08] zu erstellen. Um den neu erstellten „element“-Knoten zu verwenden muss dieser manuell über ein „segment“ mit einem anderen „element“-Knoten verbunden werden. Der „segment“-Knoten kann im SegmentsWidget erstellt werden (siehe Abschnitt 3.5.10). Da die „section“ wesentlicher Bestandteil eines Parts ist, kann sie nicht unabhängig vom Part gelöscht werden.

3.5.9 PositioningWidget

Das in Abbildung 3.32 gezeigte PositioningWidget ermöglicht den direkten Zugriff auf CPACS’ „positioning“-Knoten. Auch dieser Knoten beinhaltet Attribute wie den Namen, die uID und eine Beschreibung des Knotens. Außerdem können die Attribute des „positioning“-Knotens, Länge, Pfeilung und V-Stellung direkt bearbeitet werden. Da das PartWidget einen intuitiveren Zugriff auf die Parameter bietet, wird dieses Widget voraussichtlich nur in Ausnahmefällen Verwendung finden.

Um zu definieren, von welchem „section“-Element aus die Parameter des „positioning“ definiert sind, kann über die „From Section“ eine „section“ ausgewählt werden. Da die „To Section“ über den Part zu dem das „positioning“ gehört bereits definiert ist, wird dort lediglich der Vollständigkeit halber die Ziel-„section“ angezeigt.

Auch der „positioning“-Knoten kann nicht unabhängig vom jeweiligen Part gelöscht werden.

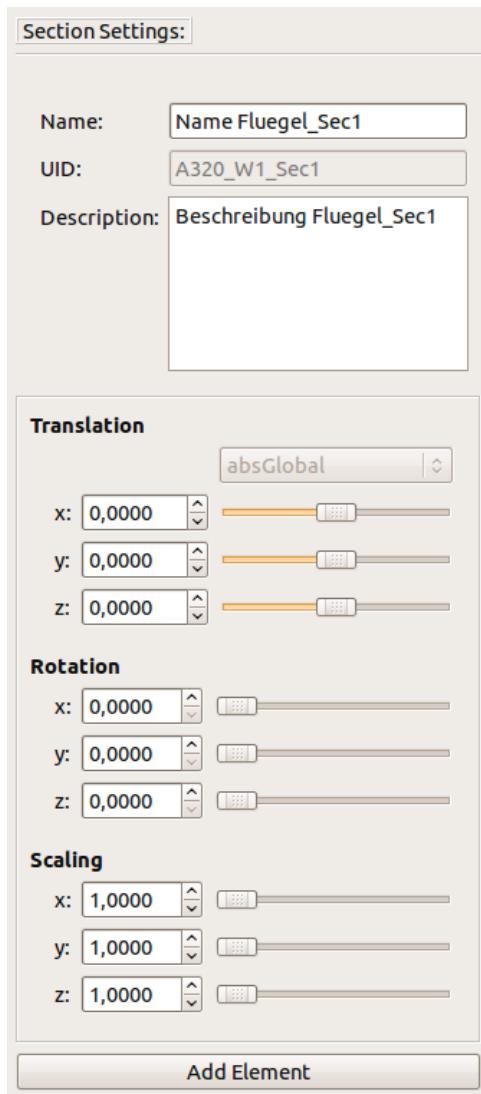


Abbildung 3.31: WingSectionWidget

3.5.10 SegmentsWidget

Das SegmentsWidget dient dazu neue „segment“-Knoten zu einem Part hinzuzufügen. Da es ansonsten keine weitere Funktion hat enthält es zu diesem Zweck lediglich einen Button.

3.5.11 SegmentWidget

Das SegmentWidget ermöglicht es CPACS' „segment“-Knoten zu bearbeiten. Abbildung 3.33 zeigt das Layout des Widgets. Da die „segments“ dazu dienen zwei „element“-Knoten zu einem Volumenkörper zu verbinden werden hier lediglich die beiden „element“-Knoten referenziert. Als „From Element“ können alle Elemente ausgewählt werden, die in anderen Parts definiert sind. Das „To Element“ kann eines der im aktuellen Part vorhandenen „element“-Knoten gewählt werden.

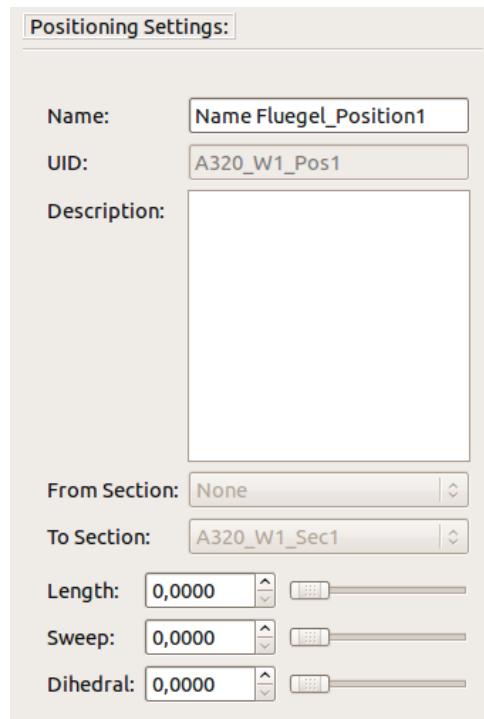


Abbildung 3.32: WingPositioningWidget

3.5.12 ElementWidget

Das ElementWidget aus Abbildung 3.34 bietet Zugriff auf die „element“-Knoten. Über dieses Widget können der Name, die uID, die Beschreibung, die Transformation, und die referenzierte Airfoil bzw. das referenzierte Rumpfprofil bearbeitet werden.

3.5.13 3 Seitenansichten

Um die drei Seitenansichten zu erstellen, werden die Randkoordinaten der Segmente ermittelt, welche die Komponente definieren. Die Berechnung der Koordinaten ist in Abschnitt 3.3.2 beschrieben. Die drei Ansichten können über die jeweiligen Menüpunkte unter **Views** aktiviert und deaktiviert werden. Zusätzlich besteht die Möglichkeit die Sichtbarkeit der Ansichten über die Funktionstasten **F2** bis **F4** zu steuern.

Die Ansichten sind durch sogenannte DockWidgets realisiert, die frei schwebend verwendet oder an der rechten und linken Seite des Hauptfensters angedockt werden können. Über die Maus können die Ansichten verschoben (drücken und halten der linken Maustaste) und Skaliert (drehen des Mausrades) werden. Sollte die Geometrie einmal nicht mehr im Fenster sichtbar sein kann sie durch drücken der Leertaste wieder im Zentrum der Ansicht platziert werden. Dabei muss das Dockwidget der Ansicht aktiviert sein und den Focus haben. Dazu genügt es mit der linken Maustaste in den Bereich der Ansicht zu klicken.

Um Flugzeugentwürfe anhand von Skizzen durchzuführen können Hintergrundbilder in die Ansichten geladen werden. Die Hintergrundbilder können im Menü unter **Menu-**

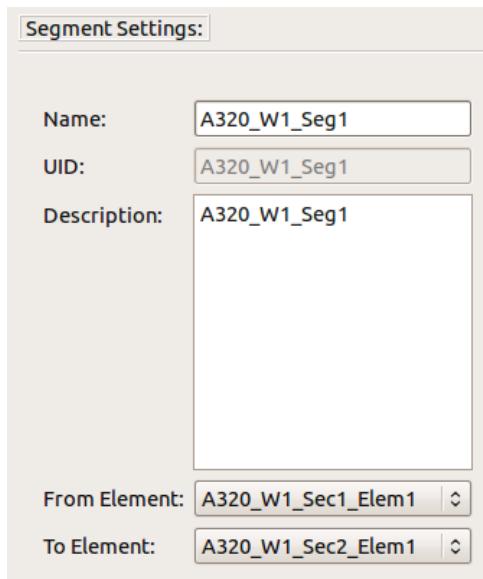


Abbildung 3.33: WingSegmentWidget

>*Preferences* konfiguriert werden. Details über die Einstellungen zu den Hintergrundbildern befinden sich in Abschnitt 3.5.14. Abbildung 3.35 zeigt die drei Ansichten welche den Flugzeugentwurf aus Abschnitt 4.1.1 zeigen.

3.5.14 Einstellungen

Die Einstellungen werden über das configWidget aus Abbildung 3.36 getätigigt, welches über das Menü unter *Menu->Preferences* erreicht wird. Der obere Teil der Einstellungen besteht aus den folgenden Elementen:

TIGLViewer Über diese Einstellung wird der Pfad zum TIGLViewer angegeben. Über den „...“-Button wird der Dateidialog gestartet mit dem die ausführbare Datei ausgewählt werden kann.

CPACS Library Dieser Punkt gibt den Pfad zu der verwendeten CPACS-Bibliothek an. Aus der Bibliothek können Profile und Geometrien geladen werden. Dieser Pfad kann ebenfalls über einen Dateidialog ausgewählt werden.

Airfoil Unter „Airfoil“ wird die uID des Standard-Tragflächenprofils angegeben. Diese wird für alle neu erstellten „Wing“-Elemente verwendet.

Profile „Profile“ gibt die uID des Standard-Rumpfprofils an. Diese wird für alle neu erstellten „Fuselage“-Elemente verwendet.

Im unteren Teil befinden sich die Einstellungen für die Hintergrundbilder der Ansichten. Die Verwendung von Hintergrundbildern wird über die jeweilige Checkbox aktiviert. Für jede der drei Ansichten können nach der Aktivierung die folgenden Einstellungen gesetzt werden:

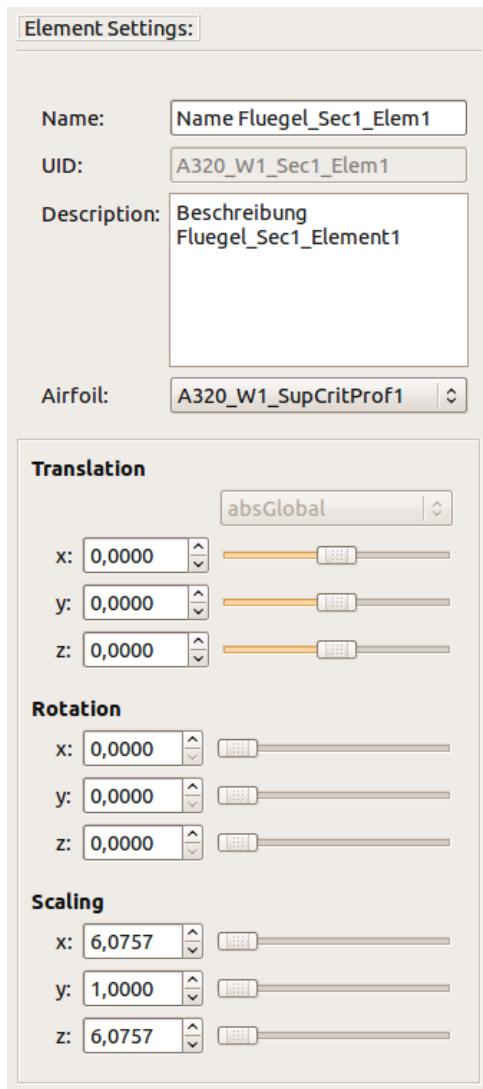


Abbildung 3.34: WingElementWidget

Image „Image“ gibt den Pfad zur Bilddatei an, die als Hintergrund für die Ansicht verwendet wird. Auch hier kann der Pfad über einen Dateidialog gesetzt werden.

Width „Width“ gibt die Breite in Pixeln an in der das Hintergrundbild dargestellt wird.

Scale Unter „Scale“ befindet sich der Maßstab des Bildes in Pixel pro Meter. Dieser Wert wird verwendet um Skizze der Ansicht auf eine zum Hintergrundbild passende Größe zu skalieren.

Getätigte Einstellungen müssen über drücken des „OK“-Buttons bestätigt werden. Beim Verlassen des Einstellungsfensters auf eine andere Weise werden die Änderungen verworfen.

Um Einstellungen zu Laden oder zu Speichern kann im Hauptmenü *Menu->Load Config* bzw. *Menu->Save Config* verwendet werden, siehe Abschnitt 3.5.1.

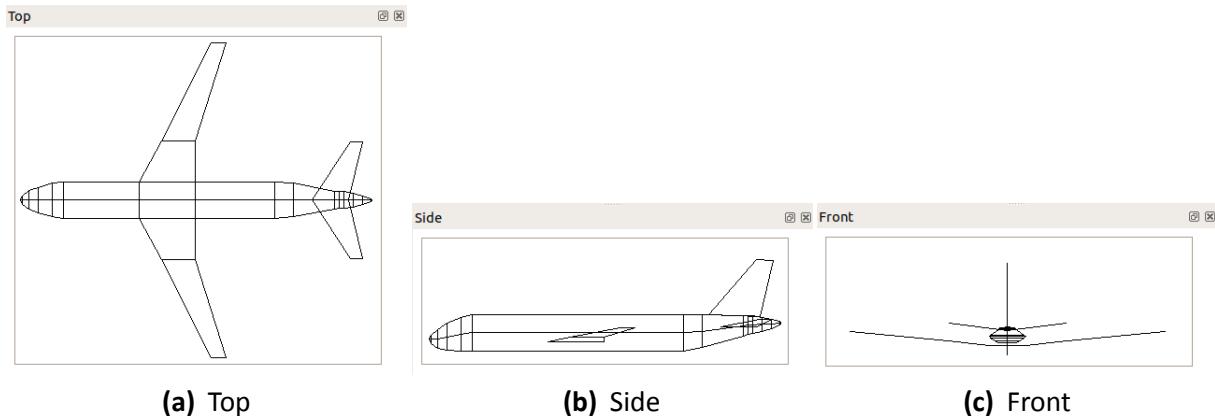


Abbildung 3.35: Drei Ansichten

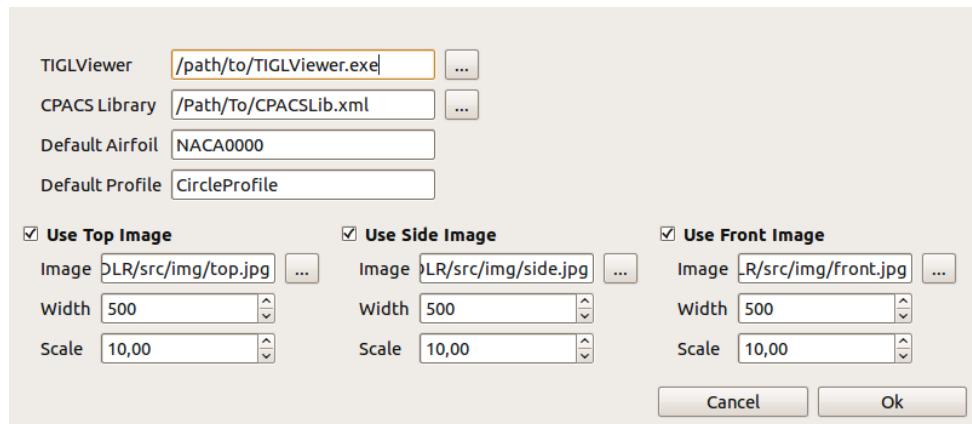


Abbildung 3.36: configWidget

3.6 TIGLViewer

Der TIGLViewer ist ein Programm welches die TIGL-Bibliothek verwendet um CPACS-Geometrien dreidimensional zu visualisieren. Die TIGL-Bibliothek wird verwendet um in CPACS gespeicherte Geometrieeinformationen zu verarbeiten. Dafür bietet TIGL eine Vielzahl an Funktionen, die über C/C++, Python, Java, und FORTRAN angesprochen werden können. TIGL ermöglicht es auch die erstellten Geometrien nach IGES/VTK zu exportieren. Für eine detailliertere Beschreibung der TIGL-Funktionalitäten wird an dieser Stelle auf die API-Dokumentation [Lit11] verwiesen.

Um neben den vereinfachten Ansichten des SGG auf eine komplette dreidimensionale Visualisierung zuzugreifen wird der TIGLViewer verwendet. Damit die Geometrie auf einfache Weise bereits während des Entwurfs auch über den TIGLViewer betrachtet werden kann, ist es möglich diesen vom SGG aus zu starten. Dazu kann entweder im Menü **Views->TIGLViewer** gewählt oder **F6** gedrückt werden.

Der TIGLViewer wird dann so gestartet, dass eine konkrete Datei geladen und auf Änderungen überwacht wird. Da das Laden der Ansicht im TIGLViewer, je nach verwendeter Hardware, bis zu wenigen Sekunden dauern kann wird die Datei nur auf Anfrage aktualisiert.

lisiert. Über das Menü unter *Views->Refresh TIGLViewer* oder drücken von **F5** werden die aktuellen Geometrien in der Datei gespeichert. Sollten im SGG mehrere Flugzeugentwürfe parallel vorhanden sein, wird nur der zur Zeit der Aktualisierung ausgewählte Entwurf übertragen. Durch die Aktualisierung auf Anfrage wird ein unnötiger Overhead vermieden.

Um den TIGLViewer zu starten muss allerdings zunächst der Pfad zur ausführbaren Datei des TIGLViewers in den Einstellungen angegeben werden.

Die Implementierung der Verbindung zum TIGLViewer ist durch die TIGLViewerThread-Klasse realisiert. Der Aufbau ist sehr einfach, da in Python eine Thread-Klasse vorhanden ist, die Prozesse in neuen Threads¹ startet. Außerdem können über die subprocess-Bibliothek externe Programme mittels „call()“ aufgerufen werden. Abbildung 3.37 zeigt die Klassendefinition der TIGLViewerThread-Klasse. Für Interessierte ist der Code der TIGLViewerThread-Klasse in Listing A.1 in Anhang A.4 aufgeführt.

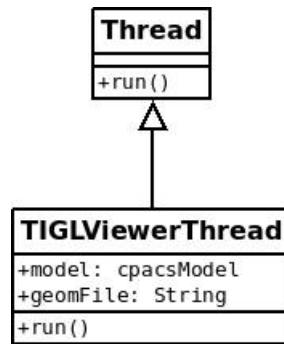


Abbildung 3.37: Klassendiagramm der TIGLViewerThread-Klasse

¹ Ein Thread ist ein Ausführungsstrang innerhalb eines Prozesses [Kai08].

4 Testen des SGG

In diesem Kapitel wird der SGG getestet. Zunächst wird in Abschnitt 4.1 gezeigt, wie mittels SGG drei unterschiedliche Flugzeugentwürfe erstellt werden können. Im Anschluss werden die erstellten Entwürfe anhand eines „Vortex Lattice Aero“-Verfahrens (Tornado) analysiert um den Output zu validieren und die Anbindung an andere CPACS-Tools zu testen.

4.1 Erstellen von Flugzeugentwürfen

Verkehrsflugzeuge haben den Zweck eine Nutzlast möglichst effizient über eine bestimmte Distanz zu transportieren. Dazu werden an den Entwurf vier fundamentale Anforderungen gestellt. Ein Flugzeugentwurf muss:

- Raum für Nutzlast (Passagiere/Fracht) bieten.
- einen Antrieb zur Fortbewegung beinhalten.
- ein Element enthalten, das Auftrieb erzeugt um das Flugzeug in der Luft zu halten.
- Steuerungsmöglichkeiten zum Manövrieren des Flugzeuges haben.

Natürlich gibt es an einen Flugzeugentwurf für ein zugelassenes Flugzeug viel mehr Anforderungen aber diese Funktionen sind die zur rein theoretischen Erfüllung der Transportleistung notwendigen Elemente.

In diesem Abschnitt werden drei bekannte Flugzeugentwürfe vorgestellt und mit dem SGG modelliert. Der erste Entwurf ist eine konventionelle Drachenkonfiguration. Als unkonventionelle Konfigurationen wurde das Konzept des Boxwings (bzw. PrandtlPlane) sowie ein moderner Blended-Wing-Body Entwurf gewählt.

4.1.1 Drachenkonfiguration

Der traditionelle Flugzeugentwurf setzt auf Funktionsdifferenzierung. Die vier Hauptfunktionen Stauraum, Antrieb, Auftrieb und Steuerung werden in voneinander getrennten Komponenten (Rumpf, Triebwerke, Tragfläche, Leitwerke) umgesetzt und können diesen eindeutig zugewiesen werden. Abbildung 4.1 zeigt die Zuweisung der Funktionen zu den Komponenten. Flugzeugentwürfe gibt es in unterschiedlichen Konfigurationen, welche sich in der Anordnung der Komponenten zueinander unterscheiden. Die bekannteste und am weitesten verbreitete Konfiguration ist die Drachenkonfiguration (Abbildung 4.2).

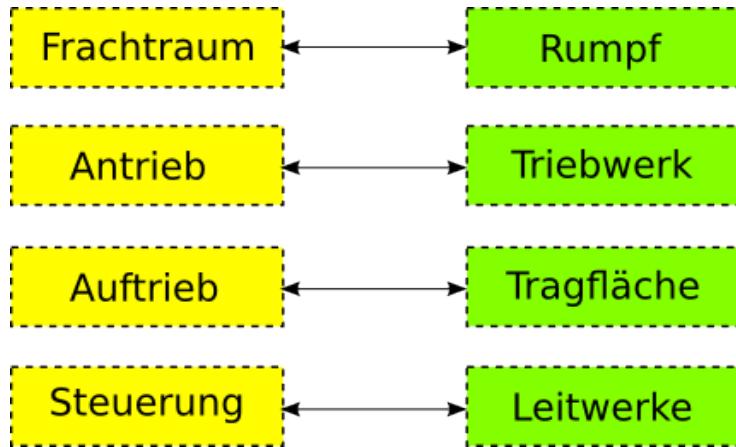


Abbildung 4.1: Zuordnung der Funktionen zu Komponenten

Bei der Drachenkonfiguration befinden sich die Leitwerke (Steuerflächen) hinter der auftrieberzeugenden Tragfläche. Das Höhenleitwerk dient der Nicksteuerung und das Seitenleitwerk der Giersteuerung. Für die Rollbewegung sind an den äußeren Enden der Tragfläche Querruder angebracht, die parallel entgegengesetzt angesteuert werden. Der zylinderförmige Rumpf bietet ein gutes Verhältnis von Stauraum zu erzeugtem aerodynamischen Widerstand. Die für den Vortrieb verantwortlichen Triebwerke sind in der Regel symmetrisch an beiden Seiten der Tragfläche befestigt.

Die Aufteilung der Funktionen auf unterschiedliche Komponenten hat den Vorteil, dass die Komplexität der einzelnen Komponenten relativ gering ist. Eine geringere Komplexität vereinfacht sowohl den Entwurf der Komponenten als auch die Fertigung. Langjährige Erfahrungen mit diesem Entwurf haben zu einer hohen Expertise und großen Optimierungen geführt. Neben der Funktionsoptimierung wurden auch die Fertigungsverfahren und -abläufe optimiert. Kriterien des Passagierkomforts sind ebenso bereits untersucht und werden im Entwurf berücksichtigt.

4.1.1.1 Entwurf mit dem SGG

Um eine traditionelle Drachenkonfiguration mit dem SGG zu erstellen wird als Vorlage eine A320 verwendet. Zum Modellieren werden drei Hintergrundbilder¹ verwendet, die in die Ansichten des SGG geladen werden. Nachdem der SGG gestartet ist, wird daher zunächst das Einstellungsmenü *Edit->Preferences* geöffnet und die Hintergrundbilder ausgewählt. Je nach Auflösung des verwendeten Monitors kann die Anzeigebreite des Bildes angepasst werden. Wie Abbildung 4.3 zeigt, wurde die Breite aller Bilder (top, side und front) für diesen Test auf 700px gestellt.

Das Bild top.png, welches die Länge des Flugzeugs darstellt ist 1018px breit ($w_{image,top} =$

¹ Als Bilder wurden schematische Skizzen aus [Air11] verwendet. Diese wurde jedoch mit einem Bildbearbeitungsprogramm vereinfacht.

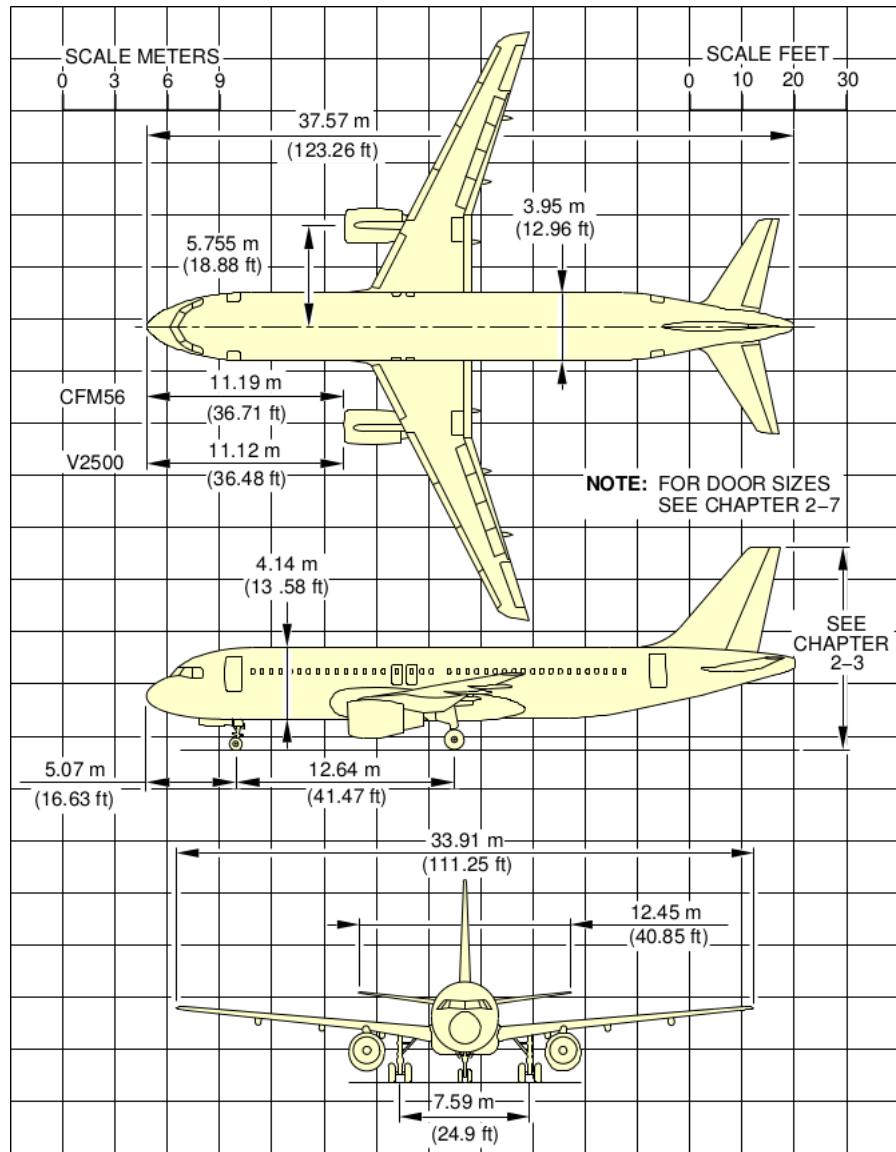


Abbildung 4.2: Drachenkonfiguration (A320) [Air11]

1018px). Anhand der Länge eines Standard A320¹ von $l_{A320} = 37.57m$ ergibt sich der Maßstab (scale) aus Gleichung 4.1.

$$scale_{A320,top} = \frac{w_{image,top}}{l_{A320}} \approx 27.10 \frac{px}{m} \quad (4.1)$$

Der Maßstab von „A320side.png“ wird analog, nur mit $w_{image,side} = 1021px$ berechnet.

$$scale_{A320,side} = \frac{w_{image,side}}{l_{A320}} \approx 27.18 \frac{px}{m} \quad (4.2)$$

¹ <http://www.airbus.com/aircraftfamilies/passengeraircraft/a320family/a320/specifications/>

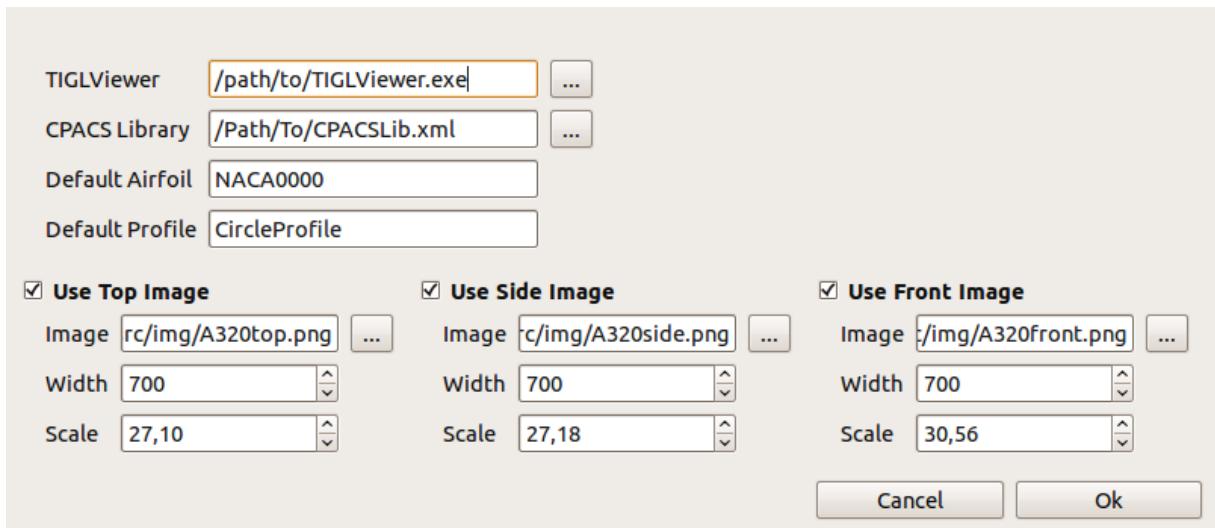


Abbildung 4.3: Einstellungen für den A320 Entwurf

Für „A320front.png“ mit einer Breite von $w_{image,front} = 1042px$ wird jedoch die Spannweite² $span = 34,10m$ benötigt. Der Maßstab berechnet sich dann aus Gleichung 4.3.

$$scale_{A320,front} = \frac{w_{image,front}}{span_{A320}} \approx 30.56 \frac{px}{m} \quad (4.3)$$

Durch die Eingabe des Maßstabs wird eine Ansicht automatisch so skaliert, dass die Längen zum Hintergrund passen. Die Eingabe des Maßstabs ist nicht notwendig, da die Skalierung auch manuell mit Hilfe des Mausrades durchgeführt werden kann.

Um die Geometrie parallel mit dem TIGLViewer zu betrachten muss der Pfad zum TIGL-Viewer unter TIGLViewerPath eingetragen werden. Der nebenstehende Button mit der Aufschrift „...“ ruft einen Dateidialog aus über den die ausführbare Datei des TIGLViewers ausgewählt werden kann.

Unter Airfoil wird „NACA0009“ als Standard-Flügelprofil eingetragen. Für die Rumpfquerschnitte wird das Kreisprofil mit der uid „CircleProfile“ verwendet. Durch Betätigen des „OK“-Buttons sind die Vorbereitungen für den folgenden Entwurf abgeschlossen.

Um die Einstellungen auch nach einem Neustart des SGG zur Verfügung zu haben sollten diese über das Menü **Menu->Save Config** gespeichert werden. Für diesen Entwurf wurden die Einstellungen unter „userdata/A320.cfg“ gespeichert.

Der Entwurf des A320 beginnt mit dem Erzeugen eines neuen Flugzeugs entweder über das Menü **Add->New Aircraft** oder durch drücken von **STRG-A**. In der Baumansicht am linken Rand erscheint daraufhin ein neuer Eintrag mit der Bezeichnung „Aircraft1“, welcher durch die linke Maustaste ausgewählt wird. In dem dann erscheinenden „AircraftWidget“ (siehe Abbildung 4.4) können der Name und eine Beschreibung eingegeben werden.

Im nächsten Schritt wird durch Drücken auf „Add Fuselage“ ein neuer **Rumpf** er-

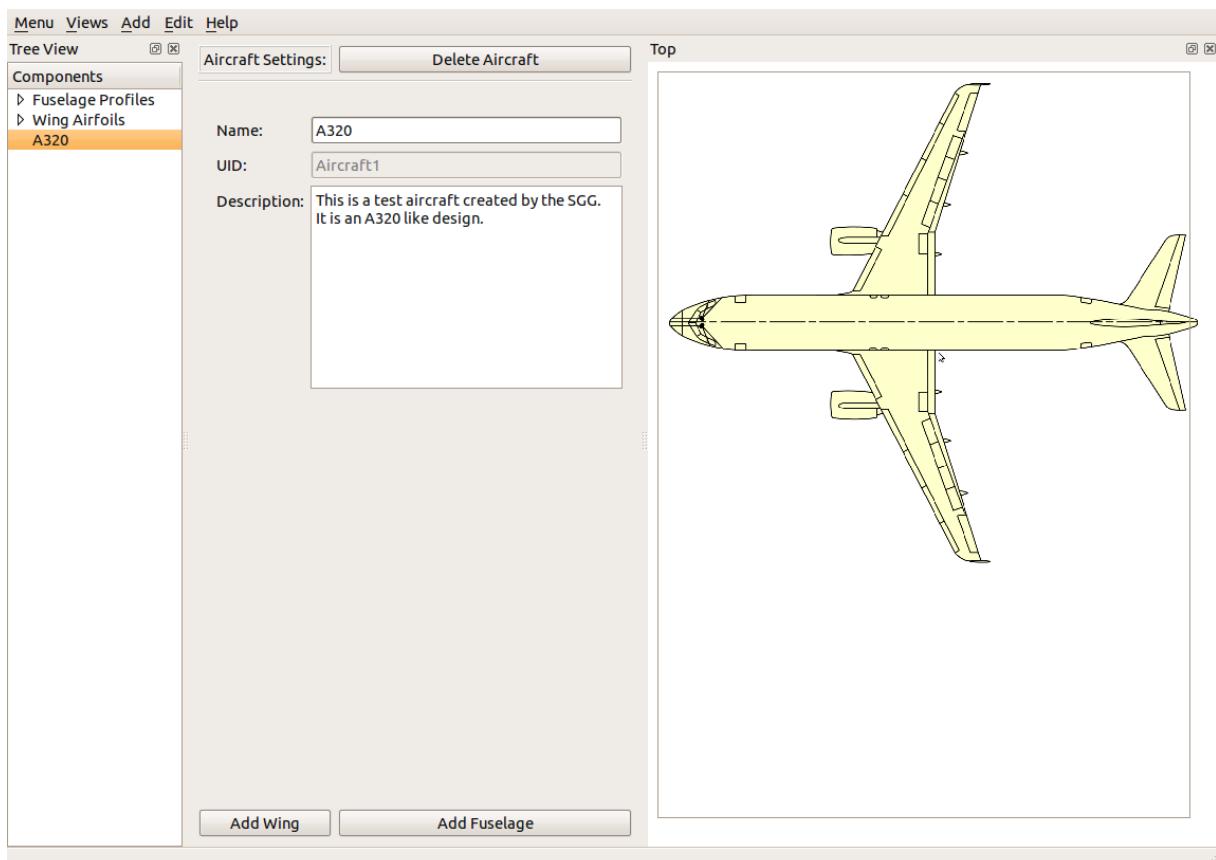


Abbildung 4.4: Erzeugen eines neuen Flugzeuges

stellt. Auch der Rumpf wird umbenannt (in „Rumpf“ und bekommt eine Beschreibung (Zylinder-Rumpf)). Der erstellte Rumpf enthält bereits einen Part. In der Ansicht ist der Rumpf aber nicht an der richtigen Position, daher wird durch drücken und halten der linken Maustaste im Ansichtsfenster und dem Bewegen der Maus der Rumpf manuell an die gewünschte Position (Rumpfnase) gesetzt. Um die Größe der Rumpfnase anzupassen wird die „RootSection“ des Rumpfes angepasst. Unter „Scaling“ werden die y- und z-Komponente auf 0,1 gesetzt. Dann werden die Rumpfparts modelliert. Durch die Auswahl des Parts in der Baumansicht¹ können die Länge und Breite des Parts bearbeitet werden. Danach kann über das FuselageWidget (Linksklick auf Rumpf) ein Querschnitt ausgewählt werden, an den ein weiterer Part angehängt wird. Da hier standardmäßig der letzte Querschnitt ausgewählt ist, muss nichts verändert werden. Durch Drücken des „Insert Part after“-Buttons wird ein neuer Rumpfpart am Ende des Rumpfes angehängt. Dieser Vorgang wird solange durchgeführt, bis der Rumpf aus der Ansicht von Oben fertig modelliert ist. Tabelle 4.1 zeigt die für die Rumpfparts verwendeten Werte.

Die Breite des zylindrischen Mittelteils von 3.95m entspricht dem Durchmesser einer

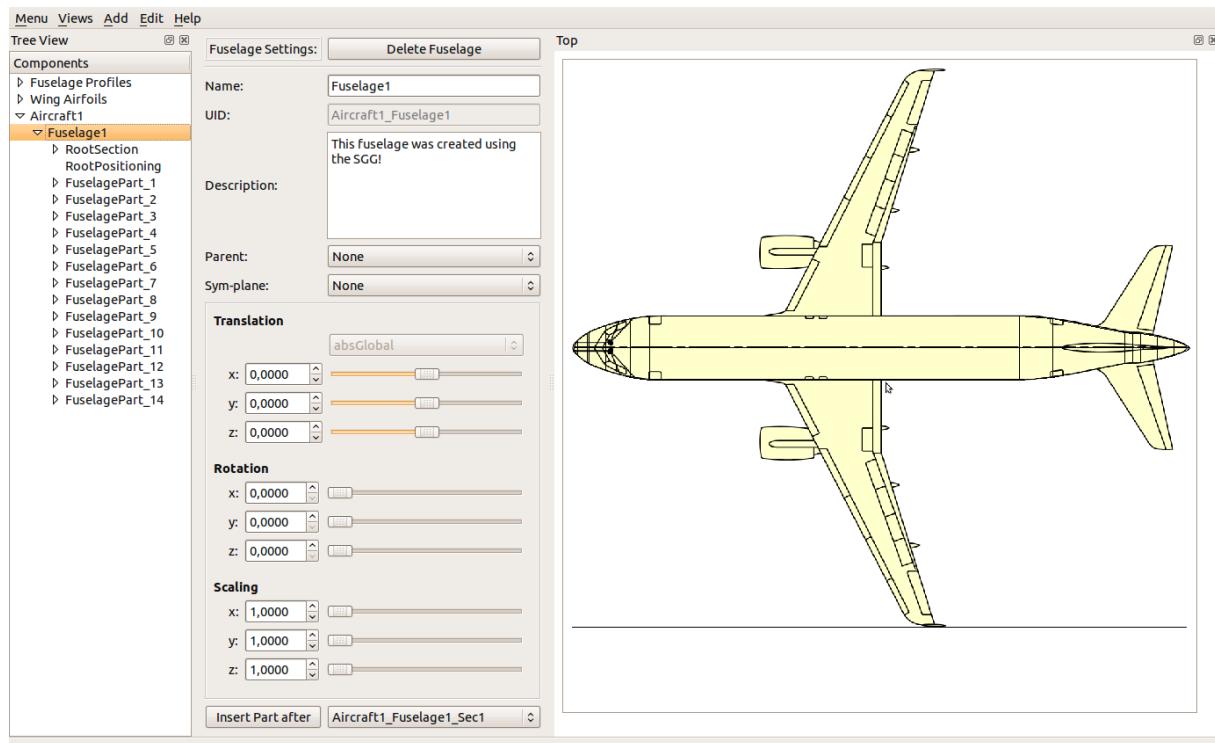
¹ Das Fuselage-Item (Rumpf) kann durch einen Linksklick mit dem Cursor auf den nebenstehenden Pfeil aufgeklappt werden.

	Länge [m]	Breite [m]
FuselagePart_1	0.2	0.9
FuselagePart_2	0.7	2.0
FuselagePart_3	1.0	2.7
FuselagePart_4	1.5	3.6
FuselagePart_5	1.2	3.95
FuselagePart_6	22.6	3.95
FuselagePart_7	2.0	3.6
FuselagePart_8	4.4	1.9
FuselagePart_9	0.5	1.8
FuselagePart_10	0.5	1.8
FuselagePart_11	1.0	1.5
FuselagePart_12	1.0	0.9
FuselagePart_13	0.9	0.3
FuselagePart_14	0.07	0.1

Tabelle 4.1: Werte für Rumpfparts des A320

A320². Abbildung 4.5 zeigt das Ergebnis dieser Schritte.

Der Rumpf ist damit allerdings noch nicht fertig. Als nächstes wird die Seitenansicht

**Abbildung 4.5:** Schritt 1 des Rumpfentwurfs (top)

(side) verwendet um den Abstand der Querschnitte von der Rumpfmittellinie einzustellen

len. Die Draufsicht wird entweder über das Menü **Views->Top**, oder über **F2** deaktiviert und die Seitenansicht entsprechend (**Views->Top** oder **F2**) aktiviert. Zunächst wird der Rumpf so positioniert, dass die Rumpfmittellinie des Mittelteils in der Mitte des Rumpfes aus der Abbildung liegt. Abbildung 4.6 zeigt den erstellten Rumpf in der Seitenansicht. Die Abstände von der Mittellinie werden nicht über die Rumpfparts, sondern über

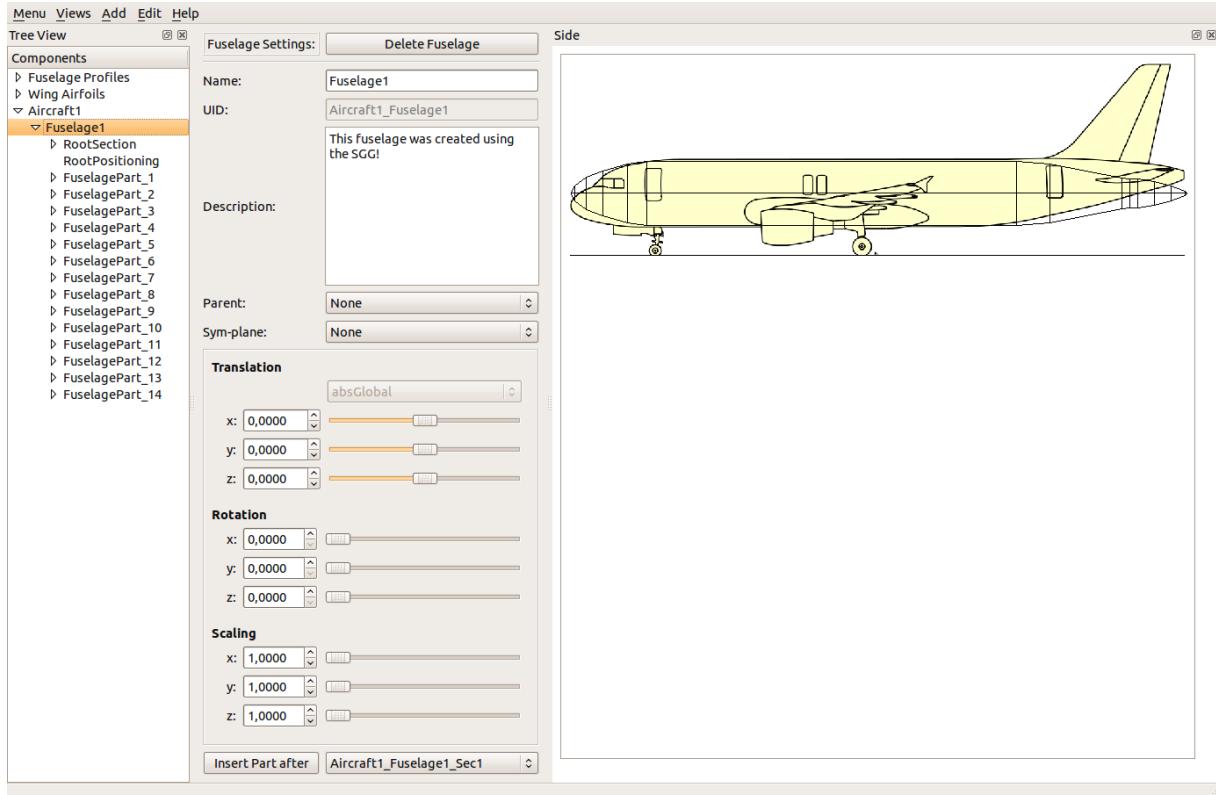


Abbildung 4.6: Schritt 1 des Rumpfentwurfs (side)

die „sections“- eingestellt. Nach dem Ausklappen eines FuselageParts kann man die jeweilige „section“ auswählen. Dort wird dann unter Translate die z-Komponente so angepasst, dass der Querschnitt in der Ansicht richtig positioniert ist. Tabelle 4.2 zeigt die in den „sections“ unter Translate angegebenen z-Komponenten der verschiedenen Rumpfparts. Nachdem alle Querschnitte richtig positioniert wurden, kann deren Höhe angepasst werden. Dazu wird zurück zu den FuselageParts gewechselt und unter „Parameter“ die Parameterkombination „Set B“ ausgewählt. Dadurch wird die Höhe zum individuellen Bearbeiten (unabhängig von der Breite) freigegeben. Tabelle 4.3 zeigt die Querschnittshöhen für die Querschnitte bei denen die Höhe angepasst wurde. Damit ist der Entwurf des Rumpfes beendet und kann im TIGLViewer betrachtet werden. Sofern der Pfad in den Einstellungen korrekt gesetzt wurde, wird der TIGLViewer entweder über das Menü **Views->TIGL-Viewer** oder über Drücken von **F6** gestartet. Im TIGLViewer werden von symmetrischen Komponenten nur die ursprünglichen und nicht die gespiegelten Geometrien dargestellt. Abbildung 4.7 zeigt die Rumpfgeometrie im TIGLViewer. Durch einen Bug im TIGLViewer werden die Querschnitte in Abhängigkeit der Skalierung

	Abstand [m]
RootSection	-0.70
FuselagePart_1	-0.70
FuselagePart_2	-0.70
FuselagePart_3	-0.43
FuselagePart_4	-0.15
FuselagePart_5	0.00
FuselagePart_6	0.00
FuselagePart_7	0.19
FuselagePart_8	0.78
FuselagePart_9	0.84
FuselagePart_10	0.86
FuselagePart_11	0.92
FuselagePart_12	0.93
FuselagePart_13	1.02
FuselagePart_14	1.02

Tabelle 4.2: Abstand der Querschnitte von der Rumpfmittellinie (A320)

	Höhe [m]
FuselagePart_3	2.862
FuselagePart_7	3.492
FuselagePart_8	2.185
FuselagePart_9	1.980
FuselagePart_10	1.890
FuselagePart_12	0.990
FuselagePart_13	0.420

Tabelle 4.3: Querschnittshöhen im Rumpf (A320)

verschoben. Daher wird der Abstand der Querschnitte von der Rumpfbezugsachse nicht korrekt dargestellt.

Um die verrichtete Arbeit zu sichern wird die bisher generierte Geometrie in einer CPACS-Datei gespeichert. Der Speichervorgang kann im Menü unter **Menu->Save File** oder über die Tastenkombination **STRG-S** gestartet werden.

Als nächstes wird die **Tragfläche** erstellt. Für die Modellierung der Tragfläche werden zwei Ansichten parallel verwendet¹. Durch Drücken von **F3** wird die side-Ansicht deaktiviert und über **F2** und **F4** die top- und front-Ansicht aktiviert. In der Baumstruktur

¹ Je nach Bildschirmauflösung und Vorliebe des Benutzers können die Ansichten parallel verwendet werden, oder über die Funktionstasten zwischen ihnen gewechselt werden.

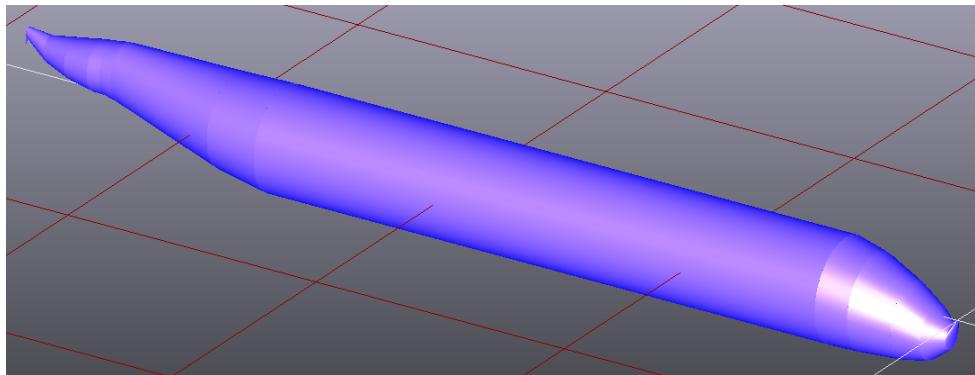


Abbildung 4.7: A320 Rumpf im TIGLViewer

tur wird dann der Flugzeugentwurf ausgewählt und über den „Add Wing“-Button eine Wing-Komponente erstellt. Diese erscheint wiederum in der Baumstruktur als „Wing1“. Durch das Auswählen der Komponente wird das zugehörige Fenster mit den Einstellungen gezeigt. Als erstes wird der Name von „Wing1“ in „Tragfläche“ geändert. Danach wird die Tragfläche positioniert indem die Translation in x-Richtung auf $12.7m$ und in z-Richtung auf $-1.0m$ gesetzt wird. Über die RootSection der Tragfläche wird die Wurzel der Tragfläche bearbeitet und mittels Skalierung der x-Achse auf 6 gestellt. Durch die Auswahl von „WingPart1“ kann der erste Part der Tragfläche modelliert werden. Da die Breite des Rumpfs am Tragflächenanschluss bereits bekannt ist, wird die Spannweite auf die halbe Rumpfbreite ($span = 1.975m$) gesetzt. Das Setzen der Zuspitzung („Taper Ratio“) auf $\tau = 1$ schließt die Bearbeitung des ersten Parts ab. Neue WingParts werden analog zu den Rumpfparts über die jeweilige Komponente erstellt. Daher wird zu „Tragfläche“ gewechselt und über den „Insert Part after“-Button gleich zwei weitere WingParts angehängt. Die neuen Parts werden dann anhand der Hintergrundbilder konfiguriert. Tabelle 4.4 zeigt die verwendeten Werte aller drei Parts. Für die zweite Seite

	Sweep [°]	Dihedral [°]	Span [m]	Taper Ratio [-]
WingPart_1	0.0	0.0	1.975	1.00
WingPart_2	27.5	7.0	4.40	0.61
WingPart_3	27.0	5.4	10.52	0.42

Tabelle 4.4: Tragflächenparameter

wird die Symmetrie der Tragfläche (Wechsel zur Tragfläche über die Baumansicht) auf „x-z-plane“ gesetzt.

Beim Erstellen des Höhen- und Seitenleitwerks wird analog zur Tragfläche vorgegangen. Sowohl das Höhen, als auch das Seitenleitwerk bestehen aus nur einem WingPart. Für das **Höhenleitwerk** werden die folgenden Werte verwendet:

- Positionierung: $x = 31.2m$, $z = 0.63m$
- RootSection Scaling: $x = 3.9$

- Pfeilung = 33.0°
- V-Stellung = 7.4°
- Spannweite = $6.25m$
- Zuspitzung = 0.32
- Symmetrie = x-z-plane

Die Werte für das **Seitenleitwerk** sind:

- Rotation des Seitenleitwerks um die x Achse um 90°
- Rotation um z = -2°
- Positionierung: x = $29.9m$, z = $1.9m$
- RootSection Scaling: x = 5.45
- Pfeilung = 38.5°
- V-Stellung = 0.0°
- Spannweite = $6.12m$
- Zuspitzung = 0.33
- Symmetrie = None

Nach diesen Schritten ist der Entwurf des A320 mit dem SGG komplett und kann über **STRG-S** gespeichert werden. Der gespeicherte Entwurf befindet sich auf der dieser Arbeit beiliegenden CD unter „SGG/src/userdata/A320.xml“. Abbildung 4.8 zeigt den fertigen Entwurf im TIGLViewer.

4.1.2 Boxwing

Der Boxwing ist ein Prinzip, bei dem ein geschlossener Tragflügel verwendet wird. Man könnte es allerdings auch als zwei miteinander verbundene Tragflächen betrachten. Abbildung 4.9 zeigt einen Boxwing Entwurf wie er in [Fre05] beschrieben wird. Der Vorteil des Boxwing ist eine Reduktion des induzierten Widerstandes, die aus der Vermeidung/Verminderung der Wirbelschleppen an der Tragflächenspitze resultiert. Die Verwirbelungen an der Tragflächenspitze entstehen durch den Druckausgleich von Unter- zu Oberseite und tragen bei einem traditionellen Tragflügel einen erheblichen Anteil zum erzeugten aerodynamischen Widerstand bei. Durch die größere Streckung des Boxwing gegenüber einer konventionellen Tragfläche wird dieser Effekt stark reduziert. Die Verbindung der Tragflächen zu einer geschlossenen „Box“ verbessert diesen Effekt zusätzlich. Rechnerisch

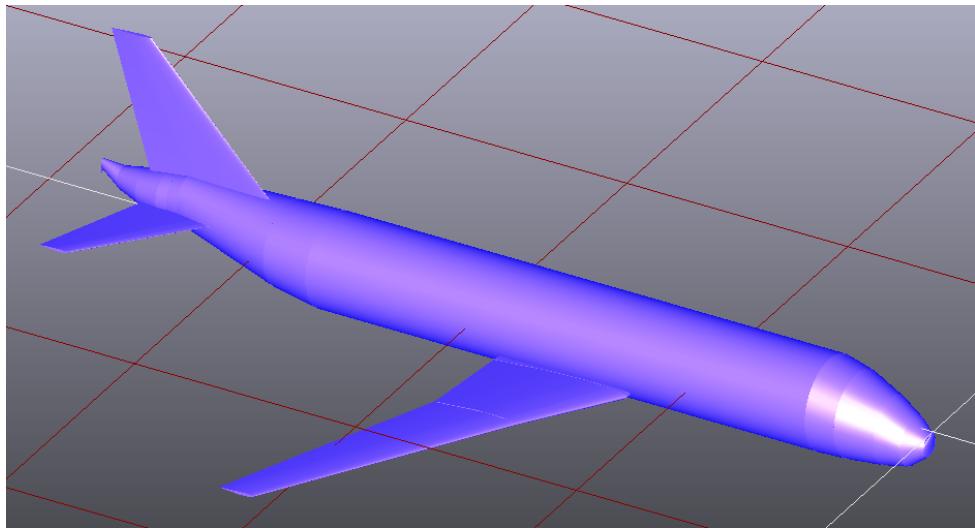


Abbildung 4.8: A320 Entwurf im TIGLViewer

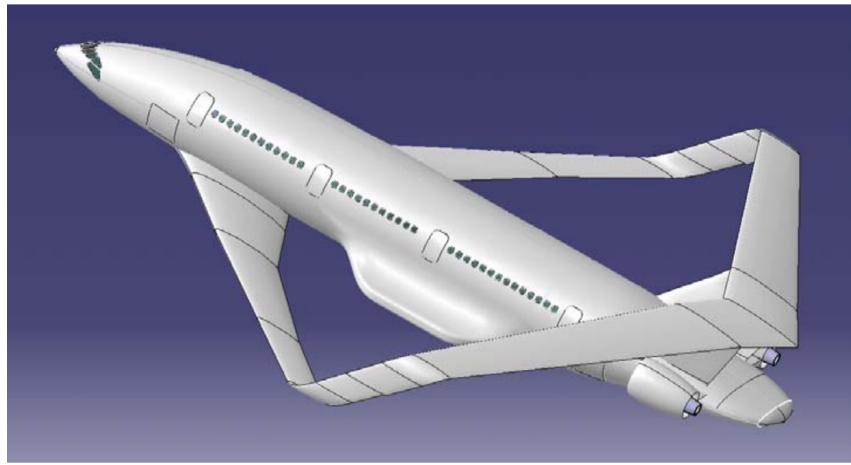


Abbildung 4.9: Boxwing (PrandtlPlane) [Fre05]

ergibt sich der Beiwert für den induzierten Widerstand aus der Tragflächenstreckung Λ , dem Auftriebsbeiwert c_A und dem Oswaldfaktor e nach Gleichung 4.4 [Eng08].

$$C_{Wi} = \frac{C_A^2}{\pi \cdot \Lambda \cdot e} \quad (4.4)$$

Bei gleichem Auftriebsbeiwert, konstantem Oswaldfaktor und größerer Streckung verringert sich somit der induzierte Widerstand.

Die größten Herausforderungen dieses Entwurfs sind die strukturelle Integration des Boxwings in den Entwurf (Rumpfanschluss) und die Unterbringung des Treibstoffes. Die strukturellen Anforderungen an einen Boxwing führen in der Regel zu einer Erhöhung des Strukturgewichts. Außerdem verringert sich durch die vergrößerte Streckung bei gleichbleibender Flügelfläche deren Volumen. Dies reduziert die Menge an Treibstoff, die in der Tragfläche untergebracht werden kann. Das bedeutet, dass mehr Treibstoff im

Rumpf untergebracht werden muss und somit weniger Platz für Nutzlast zur Verfügung steht.

4.1.2.1 Entwurf mit dem SGG

Um den Boxwing zu erstellen wird der im vorherigen Abschnitt erstellte konventionelle Entwurf geladen und bearbeitet. Die Tragfläche wird so angepasst, dass die Referenzfläche bei beiden Entwürfen gleich ist. Die Geometrie wird optisch so angepasst, dass sie dem Entwurf aus Abbildung 4.9 ähnelt.

Nach dem Laden des alten Entwurfs wird die Tragfläche als erstes so angepasst, dass die Flügelfläche halbiert wird. Die andere Hälfte der Fläche wird später für den hinteren Tragflügel benötigt. Dazu wird der Tragflügel von der Wurzel an neu konfiguriert. Das Vorgehen von der Wurzel an ist wichtig, da die Zusitzung von dem vorherigen Querschnitt abhängt. Die Referenzfläche des Tragflügels des A320 Entwurfs entspricht

$$\begin{aligned} S_{ref,A320} &= S_{ref,P1,A320} + S_{ref,P2,A320} + S_{ref,P3,A320} \\ &= 23.70m^2 + 42.50m^2 + 54.67m^2 = 120.87m^2 \end{aligned}$$

Als erstes wird daher die Profiltiefe der Tragflächenwurzel auf $x = 3.0$ skaliert und die Zusitzung des ersten WingParts auf $\tau = 1.0$ gesetzt. Das reduziert die Fläche des ersten WingParts auf $S_{ref,P1} = 11.85m^2$.

Um nun auch die Fläche des zweiten WingParts zu reduzieren wird das Parameterset „SetD“ verwendet. Darin wird dann die Pfeilung auf $\varphi = 33.0^\circ$, die V-Stellung auf $\psi = 5.0^\circ$, die Zusitzung auf $\tau = 0.68$ und die Fläche auf einen Wert von $S_{ref,P2} = 16.00m^2$ gesetzt.

Der dritte WingPart wird ebenfalls unter Verwendung von Parameterset „SetD“ angepasst. Die verwendeten Werte sind:

- Pfeilung $\varphi = 33.0^\circ$
- V-Stellung $\psi = 7.0^\circ$
- Zusitzung $\tau = 0.76$
- Fläche $S_{ref,P3} = 32.15m^2$

Somit ergibt sich die Referenzfläche der gesamten Tragfläche zu

$$\begin{aligned} S_{ref} &= S_{ref,P1} + S_{ref,P2} + S_{ref,P3} \\ &= 11.85m^2 + 16.00m^2 + 32.15m^2 = 60.00m^2 \end{aligned}$$

Als nächstes wird der Bogen des Boxwing mit den Parametern von „SetA“ modelliert. Dazu werden fünf zusätzliche WingParts an das Ende der Tragfläche angehängt und anhand der in Tabelle 4.5 gezeigten Parameter dimensioniert. Da die Spannweite entlang

	Sweep [°]	Dihedral [°]	Span [m]	Taper Ratio [-]
WingPart_4	33.0	40.0	1.27	1.00
WingPart_5	33.0	70.0	0.57	1.00
WingPart_6	33.0	90.0	0.00	1.00
WingPart_7	33.0	110.0	0.57	1.00
WingPart_8	33.0	140.0	1.27	1.00

Tabelle 4.5: Parameter des Bogens

der globalen y-Achse definiert ist, hat der WingPart_6 keine Spannweite. Um dennoch eine Länge zu vergeben, kann diese im PositioningWidget direkt angegeben werden. Für WingPart_6 wurde eine Länge von $l = 1.30m$ angegeben. Die Flächen der Bogenelemente werden für diesen Entwurf vernachlässigt.

Als letztes wird die hintere Tragfläche erstellt indem zwei weitere WingParts an die Tragfläche angehängt werden. Um die Parts anhand der Fläche zu dimensionieren wird erneut „SetD“ ausgewählt. Die vorläufigen Parameter für den hinteren Teil der Tragfläche sind in Tabelle 4.6 dargestellt. Diese Werte sind nur vorläufig, da der letzte Querschnitt

	Sweep [°]	Dihedral [°]	Span [m]	$S_{ref} [m^2]$
WingPart_9	33.0	173.0	1.5	40.0
WingPart_10	9.0	179.0	1.44	20.0

Tabelle 4.6: Parameter des Hinteren Tragflügels

noch in der xz-Ebene positioniert werden muss, um dort mit der symmetrischen Geometrie verbunden zu werden. Daher wird die Spannweite von WingPart_10 entsprechend angepasst. Da der Beginn und das Ende des Bogens, aufgrund der Symmetrie, auf der selben Spannweitenkoordinate liegen, müssen lediglich die Summe der Spannweiten von WingPart_1 bis WingPart_3 mit der von WingPart_9 und WingPart_10 verglichen werden. Anhand von Gleichung 4.5 wird die benötigte Spannweite von WingPart_10 berechnet.

$$\begin{aligned} span_{P10} &= span_{P1} + span_{P2} + span_{P3} - span_{P9} \\ &= 3.95m + 6.35m + 17.91m - 20.64m = 7.57m \end{aligned} \quad (4.5)$$

Durch die Änderung der Spannweite unter Verwendung von „SetA“ ändert sich die Fläche von WingPart_10 auf $S_{ref,P10} = 21.48m^2$.

Um den Entwurf abzuschließen, müssen nun noch das Höhenleitwerk gelöscht und das Seitenleitwerk zu einem V-Leitwerk geändert werden. Zum Löschen des Höhenleitwerks wird es in der Baumansicht ausgewählt und der „Delete Wing“-Button gedrückt.

Um aus dem Seitenleitwerk ein V-Leitwerk zu machen wird zunächst die Symmetrie auf die „x-z-plane“ gestellt und das Leitwerk um die x-Achse auf einen Winkel von 44.0° rotiert. Die Rotation um die z-Achse bleibt unverändert bei -2.0° . Die Position

des Leitwerks wird ebenfalls angepasst wobei die Translation in x-Richtung auf $27.90m$ und in z-Richtung auf $1.88m$ gesetzt wird. Als nächstes wird noch die Skalierung der RootSection in x-Richtung auf 4.45 reduziert und für den WingPart die folgenden Werte eingestellt:

- Pfeilung $\varphi = 26.5^\circ$
- V-Stellung $\psi = 0.0^\circ$
- Spannweite $span = 10.3m$
- Zuspitzung $\tau = 0.45$

Zum Abschluss wird der Entwurf über [STRG-S](#) unter „userdata/Boxwing.xml“ gespeichert und über [F6](#) mittels TIGLViewer betrachtet. Abbildung 4.10 zeigt den Boxwing Entwurf im TIGLViewer.

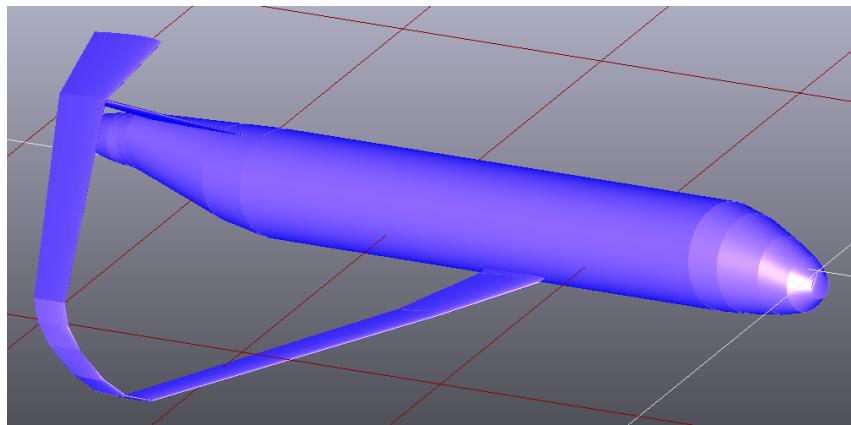


Abbildung 4.10: Boxwing Entwurf im TIGLViewer

4.1.3 Blended-Wing-Body

Das Konzept eines Blended Wing Body lässt Rumpf und Tragfläche miteinander in einer Komponente verschmelzen. Abbildung 4.11 zeigt das am DLR entwickelte Konzept eines Blended Wing Bodys. Mit einem Blended Wing Body können wesentlich mehr Passagiere befördert werden, als mit konventionellen Flugzeugen [Aircraft Design Projects; Jenkinson]. Die Integration des Rumpfes in die Tragfläche stellt außerdem ein deutlich besseres aerodynamisches Design dar mit dem im Vergleich zu konventionellen Entwürfen ca. 20% Treibstoff eingespart werden kann [Deu11]. Der BWB profitiert auch vom Beitrag des „Rumpfes“ zum Auftrieb [Kun10]. Da der Auftrieb den Gewichtskräften direkt dort entgegenwirkt wo diese auftreten, sinkt die Belastung auf die Struktur und das Flugzeug kann leichter ausgeführt werden [Mül06].

Nach [Tor82] kann ein BWB bei gleicher Spannweite und trotz deutlich geringerer Streckung ein ähnliches Gleitverhältnis haben wie ein konventioneller Entwurf. Der erhöhte

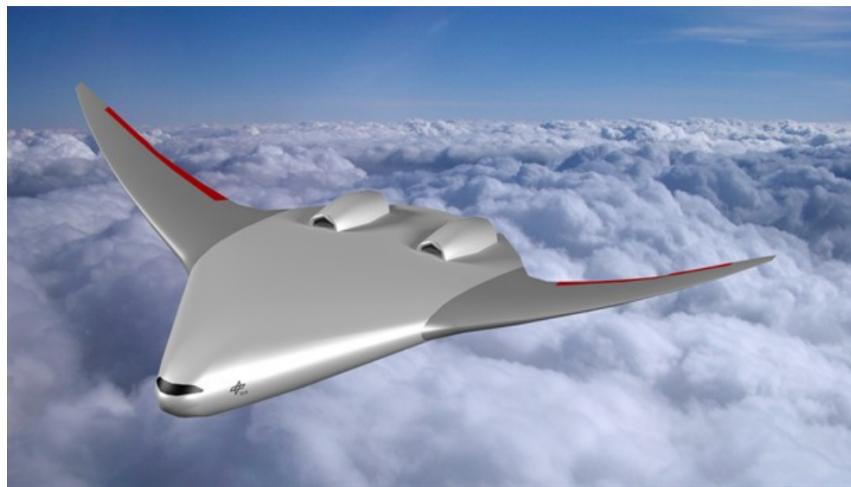


Abbildung 4.11: Blended Wing Body [Deu11]

induzierte Widerstand aufgrund der geringen Streckung kann durch eine geringere Oberflächenreibung wettgemacht werden. Somit ermöglicht das Konzept des BWBs sowohl ein großes Gleitverhältnis (Aerodynamik), als auch eine, aus struktureller Sicht, angemessene Streckung. Ein späterer Strömungsabriss, der durch zusätzlichen Wirbelauftrieb („Vortex Lift“) verursacht wird, ermöglicht der BWB den Betrieb bei größeren Angriffswinkeln [Kun10].

Zu den Herausforderungen des BWB gehört z.B. das Einbringen von Fenstern in die Kabine sowie das Sicherstellen ausreichender Evakuierungsmöglichkeiten für die Passagiere [Jen03]. Außerdem sind durch den breiten „Rumpf“¹ ein Teil der Passagiere deutlich weiter von der Flugzeugmittelachse entfernt. Das kann im Kurvenflug durch die Rollbewegung eine zusätzliche Belastungen für die Passagiere bedeuten.

4.1.3.1 Entwurf mit dem SGG

Der Entwurf des Blended Wing Bodys wird wiederum anhand der isometrischen Ansichten des BWB-Entwurfs aus [Cia11] erstellt. Die Ansichten werden wie in Abschnitt 4.1.1.1 über die Einstellungen unter **Menu->Preferences** in den SGG geladen. Um den Maßstab der Bilder zu ermitteln, wird die Wurzeltiefe des BWB-Entwurfs verwendet. Die Wurzeltiefe beträgt $l_{BWB} = 48.0m$. Die Maßstäbe der „top“ und „side“-Ansicht ergeben sich aus den Gleichungen 4.6 und 4.7.

$$scale_{BWB,top} = \frac{w_{image,top}}{l_{BWB}} = \frac{440px}{48.0m} \approx 9.17 \frac{px}{m} \quad (4.6)$$

$$scale_{BWB,side} = \frac{w_{image,side}}{l_{BWB}} = \frac{670px}{48.0m} \approx 13.96 \frac{px}{m} \quad (4.7)$$

¹ Der Teil des BWB in dem die Passagiere untergebracht sind.

$w_{image,top}$ und $w_{image,side}$ sind dabei nicht die komplette Breite des Bildes sondern nur die Anzahl der Pixel zur Darstellung der Wurzeltiefe im jeweiligen Bild. Für die Frontansicht wird die Spannweite des BWB-Entwurfs benötigt. Diese kann anhand von Gleichung 4.8 aus dem Bild für die „top“-Ansicht berechnet werden.

$$span_{BWB} = \frac{h_{image,top}}{scale_{BWB,top}} = \frac{783px}{9.17\frac{px}{m}} \approx 85.39m \quad (4.8)$$

Wobei $h_{image,top}$ die Höhe des entsprechenden Bildes in Pixel ist. Der Maßstab der „front“-Ansicht ergibt sich dann aus Gleichung 4.9.

$$scale_{BWB,front} = \frac{w_{image,front}}{span_{BWB}} = \frac{1190px}{85.39m} \approx 13.94\frac{px}{m} \quad (4.9)$$

Außerdem wird für den Blended Wing Body als Standard-Airfoil die Airfoil mit der uID „BWB_Root“ verwendet. Die Einstellungen sind unter userdata/BWB.cfg gespeichert. Als nächstes wird die RootSection in x- und z-Richtung auf 48.0 skaliert und über die „side“-Ansicht um den benötigten Anstellwinkel (hier 2.0°) um die y-Achse rotiert. Bei der Nachmodellierung des Blended Wing Bodys wird Segment für Segment vorgegangen. Jedes Segment wird in allen drei Ansichten betrachtet, um es korrekt zu dimensionieren. Dabei wird zunächst die Spannweite so gewählt, dass die Geometrie des Entwurfs ausreichend genau modelliert¹ werden kann. Danach wird die Pfeilung φ und die Zusitzung τ in der „top“-Ansicht eingestellt. Dann wird in der „side“-Ansicht die Profilverwindung und in der „front“-Ansicht die V-Stellung angepasst. Als Letztes werden die Pfeilung und Zusitzung noch einmal in der „top“-Ansicht korrigiert, da sich Änderungen der V-Stellung und der geometrischen Profilverwindung auf die „top“-Ansicht auswirken. Diese Schritte werden für jedes Segment des Blended Wing Bodys durchgeführt. Die in dieser Arbeit für den Entwurf verwendeten Werte sind in Tabelle 4.7 aufgeführt. Abbildung 4.12 zeigt das Ergebnis im TIGLViewer.

¹ Die Genauigkeit richtet sich nach den Anforderungen an den Entwurf und sollte vom Benutzer sinnvoll gewählt werden.

	Sweep [°]	Dihedral [°]	Twist [°]	Span [m]	Taper Ratio [-]
WingPart_1	15.0	1.0	0.0	2.000	0.995
WingPart_2	35.0	1.0	0.0	1.638	0.987
WingPart_3	57.0	1.0	0.0	1.731	0.968
WingPart_4	64.0	1.0	0.0	4.000	0.898
WingPart_5	64.0	1.1	0.0	2.500	0.926
WingPart_6	64.0	1.1	0.0	4.000	0.857
WingPart_7	65.0	1.1	0.0	3.856	0.825
WingPart_8	65.0	1.1	0.0	5.000	0.704
WingPart_9	65.0	1.9	-0.4	2.650	0.760
WingPart_10	34.0	3.4	0.6	2.165	0.870
WingPart_11	31.0	3.6	-0.3	3.234	0.850
WingPart_12	33.0	3.2	-2.3	2.772	0.920
WingPart_13	31.2	4.7	-0.6	8.200	0.840
WingPart_14	32.9	7.0	-0.8	5.465	0.890
WingPart_15	34.2	14.0	1.3	5.931	0.890
WingPart_16	34.7	13.3	-0.7	9.000	0.810
WingPart_17	34.5	14.9	-1.3	7.901	0.820
WingPart_18	37.7	19.0	-2.3	3.842	0.870
WingPart_19	37.8	25.0	-0.8	3.000	0.870
WingPart_20	49.3	32.5	-0.8	2.373	0.730
WingPart_21	53.1	34.2	-2.8	1.899	0.700
WingPart_22	50.3	45.0	-2.6	1.500	0.600
WingPart_23	57.7	58.0	-2.6	0.564	0.650

Tabelle 4.7: Parameter des Blended Wing Bodys

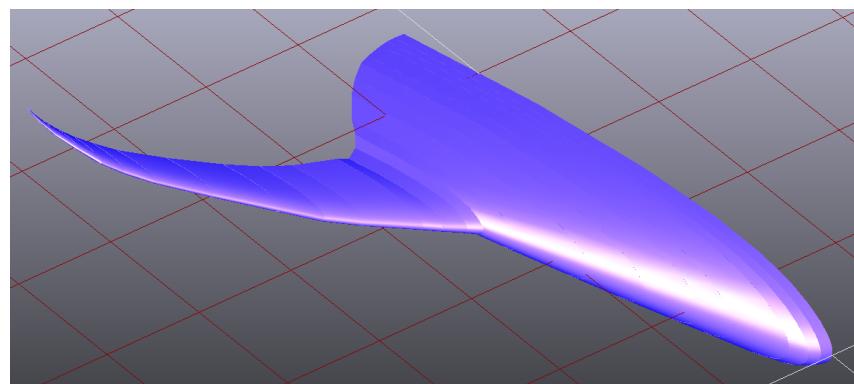


Abbildung 4.12: Blended Wing Body Entwurf im TIGLViewer

4.2 Anbindung an Tornado

Um den Export der Geometrien vom SGG nach CPACS zu testen wird das aerodynamische Analysetool Tornado verwendet, welches in Abschnitt 2.3.4 bereits vorgestellt wurde. Tornado benötigt neben den CPACS Geometrien noch die in Listing 4.1 gezeigten, zusätzlichen Informationen.

Listing 4.1: Zusätzliche Daten für Tornado

```

1 <analyses>
2   <aeroPerformanceMap>
3     <machNumber mapType="vector">0.7</machNumber>
4     <reynoldsNumber mapType="vector">0.0</reynoldsNumber>
5     <angleOfYaw mapType="vector">0.0</angleOfYaw>
6     <angleOfAttack mapType="vector">2.0</angleOfAttack>
7   </aeroPerformanceMap>
8 </analyses>

```

Diese vier Parameter (Machzahl, Reynoldszahl, Gierwinkel, Anstellwinkel) geben Tornado Informationen über den Flugzustand für den die vier Beiwerte C_A , C_W , C_{W0} und C_M berechnet werden sollen. Für alle drei Entwürfe wird eine Machzahl von 0.7, ein Gierwinkel von 0.0° und ein Anstellwinkel von 2.0° angenommen. Die Reynoldszahl wird für die Entwürfe aus Gleichung 4.10 bestimmt.

$$Re = \frac{v \cdot t_\mu}{\nu} \quad (4.10)$$

Wobei v die Geschwindigkeit der umströmenden Luft, t_μ die Bezugsflügeltiefe und ν die kinematische Viskosität der Luft ist. Die Strömungsgeschwindigkeit wird aus der Machzahl Ma und der Schallgeschwindigkeit c nach Gleichung 4.11 berechnet.

$$v = Ma \cdot c \quad (4.11)$$

Aus der US-Standardatmosphäre 1976 bei einer Höhe von $11000m$ ergeben sich die Schallgeschwindigkeit $c = 295.31 \frac{m}{s}$ und die kinematische Viskosität $\nu = 46.991 \cdot 10^{-6} \frac{m^2}{s}$. Somit ergibt sich die Geschwindigkeit als:

$$v = 0.7 \cdot 295.31 \frac{m}{s} = 206,717 \frac{m}{s}$$

Die Bezugsflügeltiefe wird für die drei Entwürfe Anhand von Gleichung 4.12 berechnet [Hei05].

$$t_\mu = \frac{span}{S_{ref}} \int_0^1 t^2(\eta) d\eta \quad (4.12)$$

η stellt dabei die dimensionslose Spannweitenkoordinate dar.

Da die Tragflächensegmente aus Trapezelementen aufgebaut sind, lässt sich die Funktion für die Profiltiefe in diese aufteilen. Für eine Tragfläche aus n Tragflächensegmenten wird die Bezugsflügelfläche somit nach Gleichung 4.13 berechnet.

$$t_\mu = \frac{span}{S_{ref}} \left(\int_0^{\eta_1} t^2(\eta) d\eta + \int_{\eta_1}^{\eta_2} t^2(\eta) d\eta + \dots + \int_{\eta_{n-1}}^1 t^2(\eta) d\eta \right) \quad (4.13)$$

Die Funktion für die Profiltiefe eines Trapezsegments (Abbildung 4.13) ergibt sich aus Gleichung 4.14.

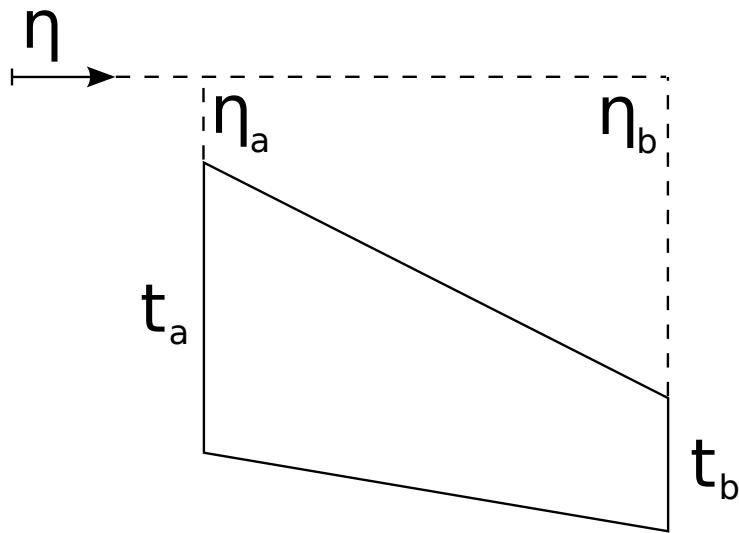


Abbildung 4.13: Tragflächensegment

$$t_{seg}(\eta) = t_a - (t_a - t_b) \cdot \frac{\eta - \eta_a}{\eta_b - \eta_a} \quad (4.14)$$

Für ein Teilsegment ergibt sich das Integral über $t^2(\eta)$ anhand von Gleichung 4.15.

$$\int_{\eta_a}^{\eta_b} t_{seg}^2(\eta) d\eta = t_a^2 \int_{\eta_a}^{\eta_b} 1 d\eta - 2 \cdot t_a \cdot \frac{t_a - t_b}{\eta_b - \eta_a} \int_{\eta_a}^{\eta_b} (\eta - \eta_a) d\eta + \left(\frac{t_a - t_b}{\eta_b - \eta_a} \right)^2 \int_{\eta_a}^{\eta_b} (\eta - \eta_a)^2 d\eta \quad (4.15)$$

Das Lösen des Integrals liefert Gleichung 4.16.

$$\int_{\eta_a}^{\eta_b} t^2(\eta) d\eta = (\eta_b - \eta_a) \cdot \left(t_a^2 - t_a \cdot (t_a - t_b) + \frac{(t_a - t_b)^2}{3} \right) \quad (4.16)$$

Diese Gleichung wird für jedes Trapezsegment berechnet und in Gleichung 4.13 eingesetzt. In Tabelle 4.8 sind die Ergebnisse der Berechnung der Bezugsflügeltiefen aufgeführt. Detaillierte Berechnungstabellen befinden sich in Anhang A.5.

Unter Verwendung von Gleichung 4.10 ergeben sich die Reynoldszahlen somit zu:

Entwurf	Spannweite [m]	Referenzflügelfläche [m^2]	Bezugsflügeltiefe [m]
A320	33.79	120.87	4.15
Boxwing	56.42 ¹	121.48	2.26
BWB	85.21	1177.91	32.41

Tabelle 4.8: Spannweiten und Referenzflügelflächen

- $Re_{A320} = \frac{206,717 \frac{m}{s} \cdot 4,15m}{46.991 \cdot 10^{-6} \frac{m^2}{s}} = 18256167 \approx 18260000$
- $Re_{Boxwing} = \frac{206,717 \frac{m}{s} \cdot 2.26m}{46.991 \cdot 10^{-6} \frac{m^2}{s}} = 9941912 \approx 9940000$
- $Re_{BWB} = \frac{206,717 \frac{m}{s} \cdot 32.41m}{46.991 \cdot 10^{-6} \frac{m^2}{s}} = 142574067 \approx 142570000$

Die vier Parameter (Machzahl, Reynoldszahl, Gierwinkel, Anstellwinkel) werden der vom SGG erstellten Datei in dem „model“-Knoten manuell (z.B. mit einem Texteditor) hinzugefügt. Um die Geometrien mit Tornado zu testen werden die Dateien in den „ToolInput“-Ordner von Tornado kopiert und die Datei des zu testenden Entwurfs in „toolInput.xml“ umbenannt. Über die Datei „mainRunTornado.m“ wird Tornado gestartet. Wenn Tornado durchgelaufen ist, befinden sich die Ergebnisse im „ReturnDirectory“-Ordner bzw. im „ToolOutput“-Ordner.

4.3 Ergebnisse

In diesem Kapitel wurde der SGG dazu verwendet drei Flugzeugentwürfe zu erstellen. Der konventionelle Entwurf der A320 anhand von Abbildungen war gut durchzuführen. Über drei Hintergrundbilder konnte die Geometrie des A320 einfach modelliert werden. Das Einfügen neuer Komponenten oder Parts ist einfach und die Parameter zum Anpassen der Geometrie intuitiv.

Auch die Variation der Tragfläche zu einem Boxwing in Abschnitt 4.1.2 war ohne Probleme möglich. Durch eine einfache Berechnung der Spannweiten konnte der hintere Teil der Tragfläche exakt bis zur Rumpfmittalachse zurückgeführt werden. In Fällen in denen die Part-Parameter unzureichend sind kann auf die Detaillierteren CPACS-Knoten direkt zugegriffen werden. So kann z.B. für einen Part dessen Längenvektor normal auf die xy-Ebene steht, und dessen Spannweite somit gleich 0 ist, über das zum Part gehörende PositioningWidget dennoch eine Länge angegeben werden.

Selbst die Modellierung eines unkonventionellen Entwurfs (Blended Wing Body) stellte kein Problem dar. Jedoch sind die Übergänge zwischen den Segmenten aufgrund der Definition in CPACS kantig.

Die unterschiedliche Herangehensweise an die Entwürfe zeigt, dass die Geometrien je nach Bedarf durch unterschiedliche Parameter definiert werden können. Die drei erstellten Geometrien wurden in CPACS-Dateien gespeichert und mit dem Analyseprogramm Tornado untersucht.

Die Analyse des A320, des Boxwing und des BWBs waren letztendlich erfolgreich. Die exportierten Geometrien wurden von Tornado akzeptiert und lieferten plausible Ergebnisse. Tabelle 4.9 zeigt die Ergebnisse der drei Entwürfe. Doch der Boxwing-Entwurf führt

	A320	Boxwing	BWB
Auftriebsbeiwert C_A	0.18841	0.14439	0.12462
Widerstandsbeiwert C_W	0.0014287	0.0015114	0.0018923
Nullwiderstandsbeiwert C_{W0}	0.008355	0.0072059	0.0058319
Momentenbeiwert C_M	-0.095282	0.38727	0.011148

Tabelle 4.9: Ergebnisse der Analyse mit Tornado

bei Tornado zunächst zu einem Programmabbruch. Die Log-Datei mit der Fehlermeldung befindet sich auf der dieser Arbeit beiliegenden CD unter „TornadoData/Boxwing/ReturnDirectory/logTornadoRun“. Eine Vergleich der CPACS-Geometrie mit der in Tornado verwendeten Geometrie bot einen Hinweis auf das Problem. Abbildung 4.14 zeigt den Vergleich der beiden Geometrien. Es ist deutlich zu sehen, dass in der Tornado-Geometrie

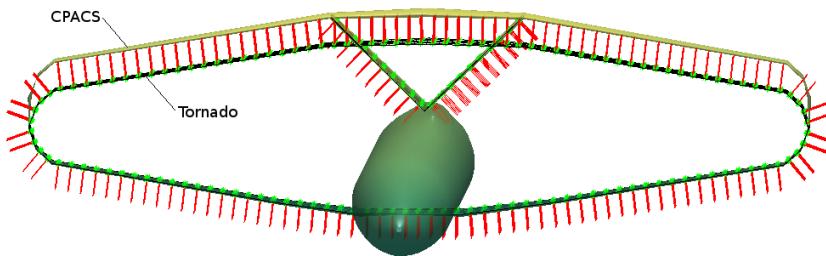


Abbildung 4.14: Unterschied in CPACS- und Tornado-Geometrie

der Teil der Tragfläche, der senkrecht zur globalen xy-Ebene steht fehlt. Bei diesem Fehler handelt es sich um einen Bug beim Generieren der Tornado-Geometrie aus dem CPACS-Datensatz. Der CPACS-Datensatz des Boxwing ist korrekt. Durch das Löschen des entsprechenden Parts in dem Boxwing und einem Anpassen der Länge des Leitwerks konnte der Bug umgangen und auch der Boxwing-Entwurf mit Tornado analysiert werden. Die von Tornado berechnete Druckverteilung, als Differenz zum Umgebungsdruck, ist in den Abbildungen 4.15 bis 4.17 dargestellt. Alle Daten zu den mit Tornado durchgeführten Tests befinden sich auf der beiliegenden CD unter „TornadoData“.

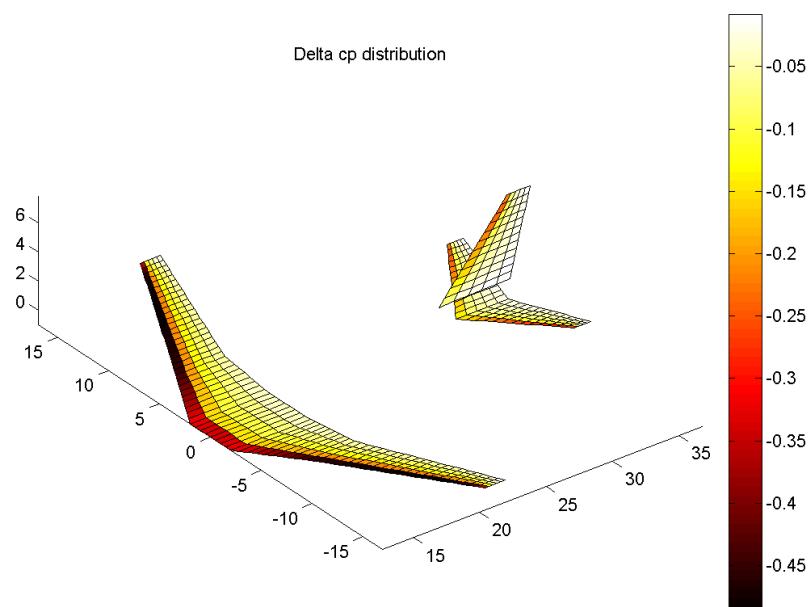


Abbildung 4.15: Druckverteilung des A320 in Tornado

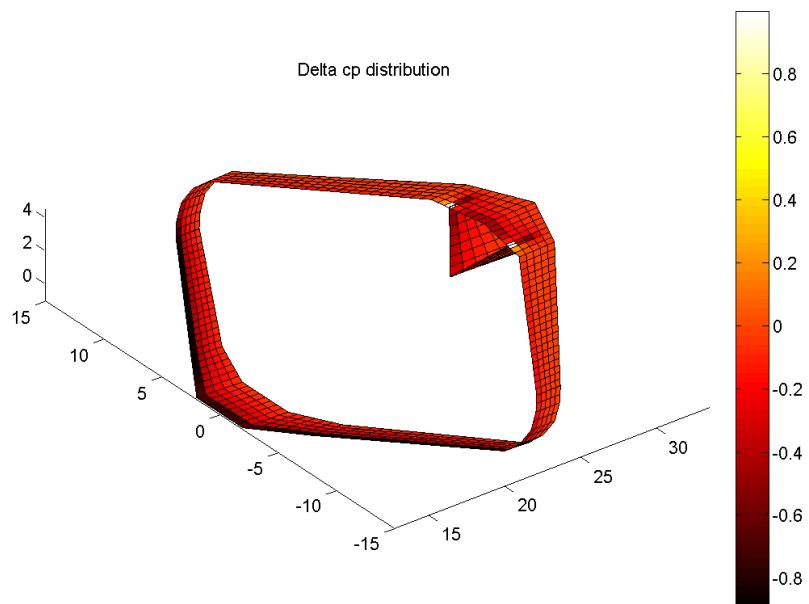


Abbildung 4.16: Druckverteilung des Boxwing in Tornado

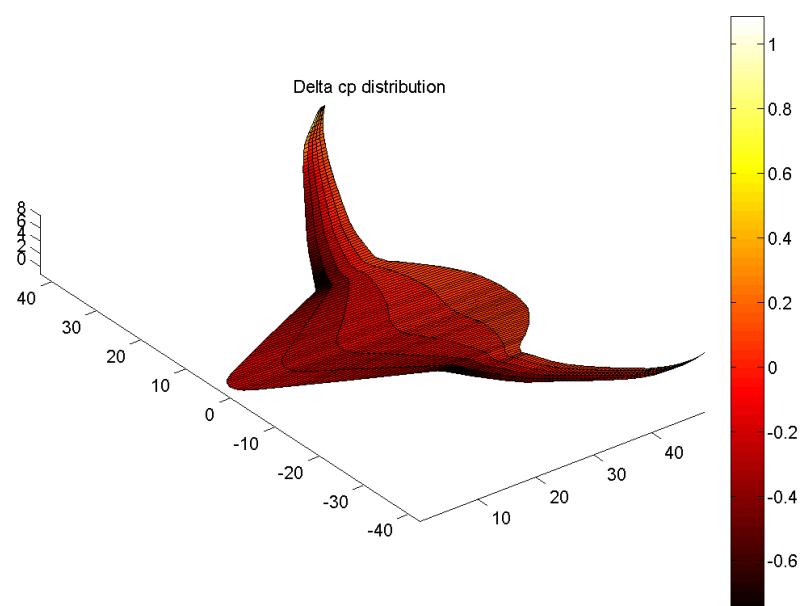


Abbildung 4.17: Druckverteilung des BWB in Tornado

5 Zusammenfassung und Ausblick

In dieser Arbeit wurde eine Möglichkeit für die Initialisierung unkonventioneller Flugzeugentwürfe in CPACS geschaffen. Der Simple Geometry Generator dient in dem Entwurfsprozess als Initiator für geometriebasierte Flugzeugentwürfe. Er ermöglicht die Generierung von konventionellen sowie unkonventionellen Flugzeugentwürfen anhand von intuitiven Geometrieparametern. Der SGG arbeitet mit CPACS-Datensätzen und erlaubt neben dem Erstellen und Speichern von Entwürfen auch das Bearbeiten bereits bestehender Entwürfe in CPACS. Dabei ist jedoch darauf zu achten, dass der SGG die CPACS Version 2.0 verwendet und sollte nur mit CPACS-Dateien dieser Version verwendet werden.

Über drei isometrische Ansichten wird die Flugzeuggeometrie visualisiert und bei Veränderungen sofort aktualisiert. Dadurch ist es möglich die Auswirkungen von Änderungen der Geometrieparameter direkt zu verfolgen.

Auch die Nachmodellierung von bestehenden Flugzeugentwürfen kann mit dem SGG einfach durchgeführt werden. Dazu werden Hintergrundbilder in die drei Ansichten geladen und die Geometrien auf Basis dieser modelliert. Durch die Angabe eines Maßstabs für die Bilder kann die automatische Skalierungsfunktion verwendet werden, welche die Ansichten auf die zu dem jeweiligen Hintergrundbild passende Größe skaliert. Die einfache Modellierung anhand von Hintergrundbildern ermöglicht es auch skizzierte Ideen für Entwürfe schnell in CPACS umzusetzen. Der Einsatz des SGG zur Erstellung von Flugzeugentwürfen auf Basis von bestehenden Ansichten wurde in Kapitel 4 beschrieben. Die Anbindung an den TIGLViewer ermöglicht es zusätzlich zu den vereinfachten isometrischen Ansichten eine dreidimensionale Ansicht des Entwurfs zu betrachten.

Im SGG vorgenommene Einstellungen können in einer Konfigurationsdatei gespeichert, und zu einem späteren Zeitpunkt wieder geladen werden. Diese Funktion ermöglicht es die Arbeit an einem Entwurf zu unterbrechen und später wieder aufzunehmen ohne die Einstellungen, wie z.B. Hintergrundbilder oder Standardprofile, erneut vorzunehmen.

Die in CPACS definierte Struktur erlaubt zur Zeit ausschließlich die lineare Verbindung von zwei Querschnitten. Daher ist auch der SGG auf diese Art der Geometrien beschränkt. Durch die Verwendung ausreichend vieler Querschnitte kann jedoch prinzipiell jede beliebige Form angenähert werden.

Beim Erstellen der Testentwürfe ist der Wunsch nach einigen Features entstanden, welche den Komfort bei der Verwendung des SGG weiter verbessern würden. Dazu gehört z.B. das Importieren von bestehenden Geometrien aus einer CPACS-Datei in einen bestehenden Entwurf im SGG. Außerdem könnte eine Profilansicht implementiert werden,

welche die Form der Profile zeigt. Diese und weitere Vorschläge befinden sich in Anhang A.3.

Bereits jetzt ist der SGG ein wertvolles Werkzeug um einen geometriebasierten Entwurf zu initiieren. Das volle Potential SGG wird allerdings erst in Verbindung mit anderen Entwurfstools deutlich. In Kombination mit weiteren CPACS-basierten Entwurfs- und Analyseprogrammen kann der SGG als Ausgangspunkt für die Untersuchung unkonventioneller Entwürfe in einem komplexen Entwurfsprozess dienen und somit einen kleinen Beitrag zu einem modernen, ganzheitlichen Entwurfssystem beitragen. Die Anbindung an Tornado zeigt beispielhaft wie die exportierten Geometrien für aerodynamische Analysen verwendet werden können. Durch die CPACS Schnittstelle können jedoch generell alle Tools verwendet werden die Geometriedaten (in der CPACS Version 2.0) verwenden. Mit einem in der verteilten Entwurfsumgebung Chameleon/RCE (aus Abschnitt 2.3.2) definierten Entwurfsprozess, wie er z.B. in [Zil11] beschrieben wird, können die vom SGG exportierten Flugzeugentwürfe in einem Vorentwurfsverfahren multidisziplinär analysiert werden.

Die Entwicklung des Simple Geometry Generators ist somit ein weiterer Schritt zur Umsetzung eines auf CPACS basierenden, flexiblen, ganzheitlichen Entwurfsprozesses.

Literaturverzeichnis

- [Adv01] ADVISORY COUNCIL FOR AERONAUTICS RESEARCH IN EUROPE: European Aeronautics: A vision for 2020 (2001)
- [Air11] AIRBUS S.A.S.: A320 airplane characteristics for airport planning (2011), revision No. 26
- [Alo04] ALONSO, Juan J.; LEGRESLEY, Patrick; VAN DER WEIDE, Edwin; MARTINS, Joaquim R. R. A. und REUTHER, James J.: pyMDO: A Framework for High-Fidelity Multi-Disciplinary Optimization (2004), AIAA
- [Ber08] BERENS, Dipl.-Ing. Martin: *Potential of Multi-Winglet Systems to Improve Aircraft Performance*, Dissertation, Technische Universität Berlin (2008)
- [Böh09] BÖHNKE, Daniel: *Data integration in preliminary Airplane Design*, Diplomarbeit, Universität Stuttgart (2009)
- [Bro11] BROUWERS, Y.H.A.: *Development of KBE applications to support the conceptual design of passengers aircraft fuselages*, Diplomarbeit, Technische Universität Delft (2011)
- [Cha12] CHAPUT, Armand J. und RIZO-PATRON, Sergio: Vehicle Sketch Pad Structural Analysis Module Enhancements for Wing Design (2012), AIAA
- [Cia11] CIAMPA, Pier Davide; ZILL, Thomas; PFEIFFER, Till und NAGEL, Björn: A Functional Shape Parametrization Approach for Preliminary Optimization of Unconventional Aircraft (2011), CEAS
- [Deu09] DEUTSCHES ZENTRUM FÜR LUFT- UND RAUMFAHRT: VAMP Projektplan (2009)
- [Deu11] DEUTSCHES ZENTRUM FÜR LUFT- UND RAUMFAHRT: Flugzeug der Zukunft - DLR forscht an Blended Wing Body (2011), URL http://www.dlr.de/dlr/presse/desktopdefault.aspx/tabcid-10307/470_read-1445/year-all
- [Deu12] DEUTSCHES ZENTRUM FÜR LUFT- UND RAUMFAHRT: CPACS 2.0 Documentation (2012)
- [Eng08] ENGMANN, Klaus (Herausgeber): *Technologie des Flugzeuges*, Vogel Buchverlag, 4. Aufl. (2008)
- [Fie99] FIELDING, John P.: *Introduction to Aircraft Design*, Cambridge University Press (1999)
- [Fre05] FREDIANI, Prof. Aldo: The Prandtl Wing (2005), Pisa University-Italy

- [Fre10] FREDERICKS, William J.; ANTCLIFF, Kevin R.; COSTA, Guillermo; DESHPANDE, Nachiket; MOORE, Mark D.; MIGUEL, Edric A. San und SNYDER, Alison N.: Aircraft Conceptual Design Using Vehicle Sketch Pad (2010), AIAA
- [Hah12] HAHN, Andrew S.: Application of Cart3D to Complex Propulsion-Airframe Integration with Vehicle Sketch Pad (2012), AIAA
- [Hei05] HEINZE, W.: *Entwerfen von Verkehrsflugzeugen II*, Braunschweig (2005), IFL-02/05
- [Int11] INTERNATIONAL AIR TRANSPORT ASSOCIATION: Vision 2050 (2011)
- [Jen03] JENKINSON, Lloyd R. und III, James F. Marchman: *Aircraft Design Projects*, Butterworth Heinemann (2003)
- [Jep10] JEPSEN, J.: Implementierung eines Geometriekonverters zur Übersetzung von Flugzeugparametrisierungen (2010), Institut für Lufttransportkonzepte, TU Hamburg-Harburg
- [Kai08] KAISER, Peter und ERNESTI, Johannes: *Python, Galileo Computing* (2008)
- [Kon10] KONING, Justin H.: *Development of a KBE application to support aerodynamic design and analysis*, Diplomarbeit, Technische Universität Delft (2010)
- [Kuh12] KUHLMAN, Dave: GenerateDS Documentation (2012), URL <http://www.rexx.com/~dkuhlman/generateDS.html>
- [Kun10] KUNDU, Ajoy Kumar: *Aircraft Design*, Cambridge University Press, Cambridge, USA (2010)
- [La 11] LA ROCCA, Gianfranco: *Knowledge Based Engineering Techniques to Support Aircraft Design and Optimization*, Dissertation, Technische Universität Delft (2011)
- [Lan11] LANGEN, T.H.M.: *Development of a conceptual design tool for conventional and boxwing aircraft*, Diplomarbeit, Technische Universität Delft (2011)
- [Lit11] LITZ, Markus: TIGL API-Documentation (2011), URL <http://tigl.sourceforge.net/Doc/index.html>
- [Mel00] MELIN, Tomas: *A Vortex Lattice MATLAB Implementation for Linear Aerodynamic Wing Applications*, Diplomarbeit, Royal institute of Technology (KTH) (2000)
- [Mel06] MELIN, Tomas: *Multidisciplinary Design in Aeronautics, Enhanced by Simulation-Experiment Synergy*, Dissertation, Royal institute of Technology (KTH) (2006)
- [Mül06] MÜLLER, Stephan: Recherche zu patentierten Flugzeugkonfigurationen (2006), Hochschule für Angewandte Wissenschaften Hamburg
- [Ray99] RAYMER, Daniel P.: *Aircraft Design: A Conceptual Approach*, AIAA, Sylmar, California, 3. Aufl. (1999)

- [Riz12] RIZZI, A.; ZHANG, M.; NAGEL, B.; BOEHNKE, D. und SAQUET, P.: Towards a Unified Framework using CPACS for Geometry Management in Aircraft Design (2012), AIAA
- [Tor82] TORENBEEK, Egbert: *Synthesis of Subsonic Airplane Design*, Delft University Press, Amsterdam, Niederlande (1982)
- [vdB09] VAN DEN BERG, Tobie: *Parametric modeling and aerodynamic analysis of multi-element wing configurations*, Diplomarbeit, Technische Universität Delft (2009)
- [vH09] VAN HOEK, Maarten: *Structural Design, Analysis and Optimization of a Lifting Surface in a Knowledge Based Engineering Environment*, Diplomarbeit, Technische Universität Delft (2009)
- [Zil11] ZILL, T.; BÖHNKE, D. und NAGEL, B.: Preliminary Aircraft Design in a Collaborative Multidisciplinary Design Environment . *AIAA Aviation Technology, Integration, and Operations (ATIO)* (2011)

A Anhang

A.1 generateDS

generateDS ist eine Python Software, geschrieben von Dave Kuhlman [Kuh12], die aus einer XML-Schemadatei eine objektorientierte Python Datenstruktur erzeugt. Diese Datenstruktur beinhalten bereits Set- und Get-Methoden¹ für die einzelnen Klassen, sowie sogenannte „parser“, die eine zur Schemadatei valide XML-Datei in ein Pythonobjekt einlesen. Dieses Pythonobjekt kann dann mittels bereitgestellter Methoden bearbeitet werden, es können z.B. Elemente hinzugefügt oder entfernt werden, und wieder in eine XML-Datei exportiert werden, die konform zum Schema ist. generateDS befand sich während des Verfassens dieser Arbeit in der Version 2.7b, welche seit Dezember 2011 unter <http://sourceforge.net/projects/generateds/> zum Download zur Verfügung steht. Die Dokumentation von generateDS kann unter <http://www.rexx.com/~dkuhlman/generateDS.html> eingesehen werden. Dort werden neben einer Einführung, Hinweise zu bekannten Problemen und Beispiele gegeben.

Durch seine Funktionalität bietet generateDS eine gute Möglichkeit für den Umgang mit CPACS-Daten und die Erzeugung von CPACS-Dateien in Python. Durch die von generateDS bereitgestellten Klassen, müssen die Klassendeklarationen für einen objektorientierten Aufbau des Geometriekonverters nicht manuell erstellt werden. Im SGG können Objekte/Instanzen der, durch generateDS erstellten, CPACS-Klassen erstellt und verwendet werden.

A.2 Dateistruktur des SGG

Um den SGG weiter Entwickeln zu können ist es wichtig zu wissen, wie die Dateien mit dem Quellcode strukturiert sind. Daher werden im Folgenden die Ordner und die wichtigsten Quellcode-Dateien kurz vorgestellt.

CPACS Dieser Ordner beinhaltet Dateien die eine direkte Verbindung zu CPACS haben.

Dazu gehören z.B.:

cpacs.py In „cpacs.py“ liegen die mittels generateDS aus der CPACS-Schemadefinition erzeugten Python-Klassen.

¹ Set- und Get-Methoden dienen dem Holen (Get) und Setzen (Set) von Klassenattributen.

cpacs_SGG.py Diese Datei beinhaltet die von den generierten Klassen (*cpacs.py*) abgeleiteten Klassen. In diesen Klassen sind zusätzliche Funktionalitäten implementiert.

cpacs_20.xsd Die Datei „*cpacs_20.xsd*“ enthält die Schemadefinition von CPACS 2.0.

cpacsModel.py Diese Datei enthält die Klasse des im SGG verwendeten, zentralen Datenmodells.

gui Im Ordner „*gui*“ befinden sich alle Widgets, die im SGG verwendet werden um Benutzerschnittstellen darzustellen.

gui/editWidgets In diesem Ordner befinden sich alle Widgets, die die Bearbeitung von CPACS Geometrien implementieren. Dazu gehören z.B. alle in den Abbildungen 3.13 bis 3.16 dargestellten Klassen.

gui/views Dieser Ordner beinhaltet die Klassen für die drei Ansichten sowie deren gemeinsame Oberklasse.

internal In dem „*internal*“-Ordner befinden sich Daten die innerhalb des SGG standardmäßig verwendet werden.

Dazu gehören z.B.:

init.xml Welche CPACS-Daten (in der Regel nur Profile) beinhaltet die beim Starten des SGG geladen werden.

SGG.cfg Eine Datei mit der Standardkonfiguration, die vom SGG beim Starten geladen wird.

ViewerData.xml Diese Datei stellt die Verbindung zum TIGLViewer dar. Diese Datei wird beim Starten des TIGLViewers ([F6](#)) an diesen übergeben und auf Änderungen überwacht.

userdata Unter „*userdata*“ werden Benutzerdateien abgelegt. Dort können Entwurfs- und Konfigurationsdateien abgelegt werden.

userdata/img Im Unterordner „*img*“ können Hintergrundbilder abgelegt werden, die in den Ansichten des SGGs verwendet werden.

A.3 Zusätzliche Features

Beim Testen des SGGs ist der Wunsch nach zusätzlichen Features aufgekommen, die nicht Teil der Anforderungsliste für diese Arbeit und in diesem Abschnitt vorgestellt werden.

Profilansicht Die Vereinfachten Ansichten des SGG stellen die exakte Form der Profile nicht dar. Um über den Namen des Profils hinaus zu verdeutlichen was für ein Profil verwendet wird, könnte eine Ansicht für die Profile implementiert werden. Diese sollte die i.d.R. zweidimensionale Profilform darstellen. Die Ansicht könnte durch die Auswahl der Profile in der Baumansicht aktiviert werden.

Importieren von Geometrien Mit dem SGG können neue Entwürfe erstellt und bestehende Entwürfe geladen werden. Um jedoch einen Entwurf zu erstellen, der aus Teilen unterschiedlicher Entwürfe besteht müssen diese manuell mit einem Editor zusammengeführt werden. Daher ist es wünschenswert im SGG eine Möglichkeit vorzusehen einzelne Teile aus CPACS-Dateien in einen bestehenden Entwurf zu integrieren.

Farbauswahl für die Ansichten Wenn die Hintergrundbilder aus ähnlichen Farben bestehen wie die vom SGG gezeichneten Linien (schwarz), sind diese nicht mehr klar zu erkennen. Eine Möglichkeit wäre Farbe der Hintergrundbilder mit einem Bildbearbeitungsprogramm zu verändern. Einfacher für den Anwender wäre es eine andere Farbe für die gezeichneten Ansichten im SGG zu wählen um diese deutlich von den Hintergrundbildern abzuheben. Daher wäre es praktisch den Einstellungen für die Ansichten eine Farbauswahl hinzuzufügen.

A.4 Listings

Listing A.1: TIGLViewerThread.py

```
from threading import Thread
2 from subprocess import call

4 class TIGLViewerThread(Thread):
5     def __init__(self, model):
6         super(TIGLViewerThread, self).__init__()
7         self.model = model
8         self.geomFile = "internal\ViewerData.xml"

10    def run(self):
11        result = call([self.model.config['TIGLViewerPath'], self.geomFile])
12        # allowing to restart thread after closing the TIGLViewer
13        super(TIGLViewerThread, self).__init__()
14        return result
```

A.5 Tabellen

Die Tabellen A.1, A.2¹ und A.3 zeigen die Auswertungen von Gleichung 4.15 für die drei getesteten Entwürfe.

	t_a [m]	t_b [m]	η_a [-]	η_b [-]	Gl. (4.15) [m^2]
WingPart_1	6.00	6.00	0.0000	0.1169	4.2083
WingPart_2	6.00	3.66	0.1169	0.3773	6.1944
WingPart_3	3.66	1.54	0.3773	1.0000	4.4425

Tabelle A.1: Integralberechnung des A320

	t_a [m]	t_b [m]	η_a [-]	η_b [-]	Gl. (4.15) [m^2]
WingPart_1	3.00	3.00	0.0000	0.0700	0.6301
WingPart_2	3.00	2.04	0.0700	0.1826	0.7234
WingPart_3	2.04	1.55	0.1826	0.5000	1.0292
WingPart_8	1.55	2.33	0.5000	0.8658	1.3954
WingPart_9	2.33	3.35	0.8658	1.0000	1.0938

Tabelle A.2: Integralberechnung des Boxwing

¹ Für den Boxwing sind die Bogenelemente „WingPart_4“ bis „WingPart_7“ vernachlässigt worden.

	t_a [m]	t_b [m]	η_a [-]	η_b [-]	Gl. (4.15) [m^2]
WingPart_1	48.00	47.76	0.0000	0.0235	53.8082
WingPart_2	47.76	47.14	0.0235	0.0427	43.3343
WingPart_3	47.14	45.63	0.0427	0.0630	43.6867
WingPart_4	45.63	40.98	0.0630	0.1100	88.1176
WingPart_5	40.98	37.94	0.1100	0.1393	45.7065
WingPart_6	37.94	32.52	0.1393	0.1862	58.3782
WingPart_7	32.52	26.83	0.1862	0.2315	40.0135
WingPart_8	26.83	18.89	0.2315	0.2902	30.9725
WingPart_9	18.89	14.35	0.2902	0.3213	8.6439
WingPart_10	14.35	12.49	0.3213	0.3467	4.5726
WingPart_11	12.49	10.61	0.3467	0.3846	5.0680
WingPart_12	10.61	9.77	0.3846	0.4171	3.3774
WingPart_13	9.77	8.20	0.4171	0.5133	7.7887
WingPart_14	8.20	7.30	0.5133	0.5775	3.8600
WingPart_15	7.30	6.50	0.5775	0.6471	3.3170
WingPart_16	6.50	5.26	0.6471	0.7527	3.6653
WingPart_17	5.26	4.32	0.7527	0.8454	2.1340
WingPart_18	4.32	3.75	0.8454	0.8905	0.7349
WingPart_19	3.75	3.27	0.8905	0.9257	0.4344
WingPart_20	3.27	2.38	0.9257	0.9535	0.2238
WingPart_21	2.38	1.67	0.9535	0.9758	0.0924
WingPart_22	1.67	1.00	0.9758	0.9934	0.0320
WingPart_23	1.00	0.65	0.9934	1.0000	0,0045

Tabelle A.3: Integralberechnung des BWB

A.6 Anforderungsliste

Anforderungsliste

Anforderung	Fertigstellung (Juni 2012)	Anmerkungen
Allgemeine Anforderungen		
Laden und Speichern von Entwürfen	<input checked="" type="checkbox"/>	in CPACS 2.0 Dateien
Auswahl zwischen verschiedenen Modellen	<input checked="" type="checkbox"/>	bei mehreren Entwürfen in CPACS
Erstellen eines neuen Entwurfs	<input checked="" type="checkbox"/>	
Erstellen neuer Komponenten (Tragfläche, Rumpf)	<input checked="" type="checkbox"/>	als CPACS „wing“ und „fuselage“-Komponenten
Bearbeiten von bestehenden Geometrien	<input checked="" type="checkbox"/>	Detaillierte Auflistung weiter unten
Löschen von Geometrien	<input checked="" type="checkbox"/>	von Komponenten sowie einzelnen Teilstücken (Parts)
Vereinfachte isometrische Ansichten	<input checked="" type="checkbox"/>	
Anbindung an den TIGLViewer	<input checked="" type="checkbox"/>	starten des TIGLViewers; aktualisieren der Geometrien
Bearbeiten von bestehenden Geometrien		
Komponentenattribute ändern	<input checked="" type="checkbox"/>	Name, Beschreibung, Unterkomponenten
Transformieren von „sections“	<input checked="" type="checkbox"/>	Rotation, Translation, Skalierung
Transformieren von „elements“	<input checked="" type="checkbox"/>	Rotation, Translation, Skalierung
Wählen von Profilen für Rumpfquerschnitte	<input checked="" type="checkbox"/>	Auswählen zwischen geladenen Profilen („element“-Knoten)
Wählen von Profilen für Tragflächenquerschnitte	<input checked="" type="checkbox"/>	Auswählen zwischen geladenen Profilen („element“-Knoten)
Bearbeiten von Rumpfsegmenten (Parts) über festgelegte Parameterkombinationen	<input checked="" type="checkbox"/>	Für die Parameterkombinationen siehe unten
Bearbeiten von Tragflächensegmenten (Parts) über festgelegte Parameterkombinationen	<input checked="" type="checkbox"/>	Für die Parameterkombinationen siehe unten
Parameterkombinationen für Rumpfsegmente		
Pfeilung, V-Stellung, Segmentlänge, Querschnittsbreite	<input checked="" type="checkbox"/>	
Pfeilung, V-Stellung, Segmentlänge, Querschnittsbreite, Querschnittshöhe	<input checked="" type="checkbox"/>	
Pfeilung, V-Stellung, Segmentlänge, Breite/Länge	<input checked="" type="checkbox"/>	
Pfeilung, V-Stellung, Querschnittsbreite, Breite/Länge	<input checked="" type="checkbox"/>	

Anforderungsliste

Parameterkombinationen für Tragflächensegmente		
Pfeilung, V-Stellung, Profilverwindung, Spannweite, Zuspitzung	<input checked="" type="checkbox"/>	
Pfeilung, V-Stellung, Profilverwindung, Zuspitzung, Streckung	<input checked="" type="checkbox"/>	
Pfeilung, V-Stellung, Profilverwindung, Zuspitzung, Referenzfläche	<input checked="" type="checkbox"/>	
Pfeilung, V-Stellung, Profilverwindung, Spannweite, Referenzfläche	<input checked="" type="checkbox"/>	