Implementierung einer generischen Starrflügelflugzeug-Simulation in C++

Abschlussbericht für das Modul Effizient Programmieren I & II

von
Jan Olucak

durchgeführt am
Institut für Aerodynamik und Gasdynamik
der Universität Stuttgart

Stuttgart, im April 2018

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Ausgangslage

Der Prozess des Flugzeugentwurfs erstreckt sich über mehrere Schleifen in denen das Flugzeug immer detaillierter Ausgearbeitet wird. Um das Leistungsvermögen früh im Entwicklungsprozess zu beurteilen, können numerische Simulationen genutzt werden. Zum einen werden mathematische Modelle benötigt, um das physikalische Verhalten von spezifischen Domänen des Flugzeuges abzubilden, zum anderen werden Parameter benötigt, um besagte Modelle zu beschreiben. Viele Parameter sind erst im Laufe des Entwurfsprozesses verfügbar. Um das Flugverhalten dennoch abzubilden, wird eine Simulation mit verschiedenen Ausbaustufen und Fehlermodellen benötigt.
Aus den meisten Ingenieursanwendungen ist MATLAB nicht mehr wegzudenken. Grund hierfür ist das breite Anwendungsspektrum und die Vielzahl an zusätzlichen Toolboxen. Dabei handelt es sich um eine proprietäre Programmiersprache die auf dem jeweiligen Rechner interpretiert wird. Trotz der vielseitigen Anwendungsmöglichkeiten benötigt MATLAB für komplexere Anwendung eine größere Laufzeit, im Vergleich zu Programmiersprachen, welche direkt in Maschinensprache übersetzt werden. Die Laufzeit ist mitunter einer sehr kritische Größe bei der Nachweisführung und Leistungsrechnung. Zudem ist eine objektorientierte Programmierung nur bedingt möglich, was den Aufbau einer generischen Simulation erschwert.

1.2 Zielsetzung

Innerhalb dieser Ausarbeitung soll eine generische Flugzeugsimulation in der Hochsprache C++ implementiert werden. Generisch bedeutet in diesem Zusammenhang, dass die Simulation in der Lage ist, verschiedene, mit dem Aircraft Designer modellierte Flugzeuge simulieren zu können. Jede Ausbaustufe des Entwurfsprozesses soll simulierbar sein, ohne dass dabei der Code verändert wird. Über Input-Files sollen verschiedene Domänen-Modelle ausgewählt werden können. Ein modularer Aufbau soll zudem eine einfache Erweiterung des Simulation gewährleisten. Um das Simulations-Framework zu verifizieren, wird die in [1] implementierte Simulation in C++ übersetzt. Der gesamte Entwicklungsprozess stützt sich dabei auf die in [2] vorgestellten Methoden und Anregungen.

1.3 Kurzvorstellung Programm Aircraft Designer

In der Arbeit nach [1] wurde eine MATLAB basierendes Programm entwickelt, mit dessen Hilfe Simulationsmodelle für Flugzeuge modelliert werden können. Die fertigen Modelle werden sowohl in Text-Files, als auch komprimierten mat-files gespeichert. Um die Modelle zu verifizieren wurde zudem eine Simulation mit 6 Freiheitsgraden implementiert. Das Programm Aircraft Designer kann jederzeit um weitere Methoden ergänzt werden, um so den Entwurfsprozess fortzusetzen. Mittels dem semi-empirischen Tool DataCompendium (DATCOM) wird die Aerodynamik des Flugzeuges für zuvor definierte Flugzustände berechnet. In einem automatisierten Prozess wird ein Zustandsregler für die Flugzustände entworfen. Ist der Entwurfsprozess abgeschlossen kann durch die zuvor erwähnte Simulation die Güte der Modelle beurteilt werden und gegebenenfalls durch Anpassung der Parameter modifiziert werden.

2 Aufbau der Simulation

2.1 Grundidee des Simulation-Frameworks

Für die Umsetzung einer generischen Simulation werden die einzelnen Domänen des Flugzeuges in Modulen implementiert. Innerhalb der Module werden die verschiedenen Ausbaustufen oder mathematischen Modelle der Domänen implementiert. Um nun einen generischen Aufruf zu gewährleisten wird eine Klasse gleichnamig zu dem jeweiligen Modul implementiert, in der zum Einen die Auswahl für das jeweilige Modell getroffen wird, zum Anderen der eigentlichen Funktionsaufruf. Die Auswahl wird über ein Input File durch Veränderung eines Parameters getroffen.
Um nun einen generischen Aufruf zu gewährleisten wird in jedem Modul einen Basis-Klasse definiert. Neue Modelle können von der Basis-Klasse selbst oder von einem Kind der Basis-Klasse erben. Wird nun über das Input-File ein Modell ausgewählt, wird über eine switch-Funktion die Basis-Klasse initialisiert, wobei der Zeiger auf das jeweilige Modell gerichtet wird.

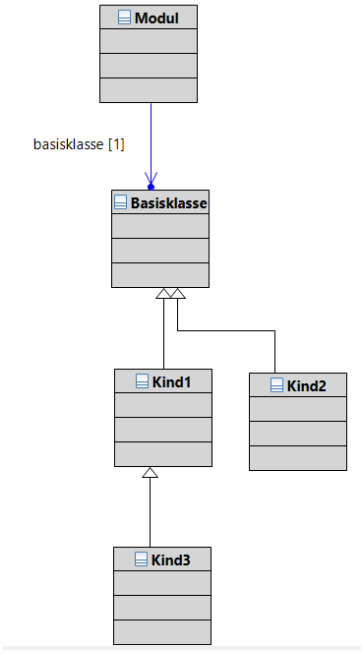

Abbildung 2.1: UML Diagramm des grundlegenden Funktionsaufruf

Somit kann das Modul jederzeit um neue Modelle erweitert werden, ohne den Funktionsaufruf selbst anzupassen.

2.2 Ablauf der Simulation

3 Durchführung der Implementierung

3.1 Entwicklungsumgebung und verwendete Tools

3.2 Dokumentation, User Manual und Coding Standard

3.3 Unit- und Modultests

3.4 Verifikation der Simulation

[3]

Literaturverzeichnis

[1] Jan Olucak. *Modellierung und Implementierung eines Starflüglermodells zur Untersuchung von Außenlasten*. Bachelorarbeit, Technische Hochschule, Ingolstadt, 2017-02-15.

[2] Manuel Kessler. Effizient programmieren i, Sommersemester 2017.

[3] Manuel Kessler. Effizient programmieren ii, Wintersemester 2017/18.