Vordiplomarbeit

Diplomanden: Widmer Christian

Tägernaustrasse 59
8645 Jona

Betreuer: Daniel Forrer

Ort, Datum: Jona, 17.02.03

Dok. Version 1.00

Inhaltsverzeichnis

1 Vorwort

Das ganze Projekt war ein grosses Stück Arbeit. Als ich damit begonnen habe, dachte ich nicht, dass es so aufwendig sein würde. Im Verlauf dieses Projektes habe ich auf allen Ebenen dazu gelernt. Nicht nur meine Programmierkenntnisse habe ich wesentlich vertiefen können, sondern auch gesehen wie wichtig eine gute Planung ist. Zu Beginn des Projekts habe ich Wahrscheinlich den Fehler gemacht, dass ich zuerst programmieren wollte und erst später die Dokumentation erstellte. Dieser Fehler hat dazu geführt, dass ich mein gestecktes Ziel von 150 Stunden überschritten habe. In der Berufswelt heisst das, dass Mehrkosten aufgelaufen wären. Zum Glück habe ich den Start des Projektes schon anfangs November gemacht. Die Zeitreserve hat mir dann ermöglicht, nochmals von vorne zu beginnen und eine saubere Planung durchzuführen. Dadurch war die Projektdurchführung strukturierter und einfacher. Ich denke, dass diese Vordiplomarbeit ein Meilenstein in meiner noch jungen Entwicklerkarriere war. Ich bin sicher, dass ich in Zukunft die gleichen Fehler nicht mehr machen werde. Neben der Planung und der Codierung war es für mich schwierig abzuschätzen, was in eine Dokumentation gehört und was nicht bzw. wie eine solche aufgebaut sein muss. Ich denke dabei, dass wir in der Schule zu wenig auf ein solches Projekt vorbereitet wurden. Es ist aber auch möglich, dass es so gewollt war, damit wir das Ganze selber erarbeiten können. Mit Hilfe von Herrn Daniel Forrer, habe ich dann auch die Unklarheiten beseitigen können. Der frühe Start in das Projekt, war nicht nur von Vorteil. Da ich der Einzige der Klasse war, der so früh mit dem Projekt begonnen hat, war es nicht möglich mit Anderen Erfahrungen auszutauschen. Zum Glück hat mir Markus Burri (C++ Lehrer der TSU) bei Problemen unter die Arme gegriffen. Zusammenfassend muss ich sagen, dass ich sehr stolz bin auf die Applikation die ich erstellt habe. Das Feedback, dass ich von den Benutzern bekommen habe, hat mir das auch bestätigt.

Jona, den 17.02.03

Widmer Christian

2 Einleitung

Das vorliegende Projekt wird anhand der Programmiersprache C++ realisiert. Die Dokumentation befasst sich zuerst mit der Projektplanung, dann wird den Leser die Anforderungsspezifikationen aufgezeigt. Nach den Voruntersuchungen wird genauer auf das Konzept und deren Implementierung eingegangen. Zum Schluss folgen dann noch die Tests und die kritische Betrachtung der Erfüllung der Anforderungen. Die vorliegende Dokumentation beschäftigt sich mit der Thematik einer Flugplanungs-Applikation. Sinn und Zweck dieser Arbeit ist es, eine Flugplanung und ein Flugbuch für den Privatpiloten zu erstellen. Diese beinhaltet zwei Teile. Zum einen die Flugplanung und zum anderen das Flugbuch. Die Flugplanung dient dem lückenlosen Erstellen eines Fluges von A nach B, das Flugbuch dem Erfassen der geflogenen Flugstunden. Aus den durch den Benutzer eingegebenen Daten soll ein druckfertiges Formular generiert werden können.

2.1 Aufgabenstellung

Erstellen Sie ein Programm welches den Privatpiloten unterstützt,

a) in der Planung eines Fluges.
   → Die benötigten Daten, für die Durchführung des Fluges, sind auf einem Formular auszudrucken.

b) im Führen des Flugbuches.
   → Die benötigten Daten, für das Bundesamt für Zivilluftfahrt, sind auf einem Formular auszudrucken.

Die Aufgabe umfasst folgende Punkte:
- Projektplanung
- Festlegung der benötigten Daten für die Flugplanung und das Flugbuch
- Evaluation einer geeigneten Programmiersprache
- Evaluation einer geeigneten Datenbank
- Entwurf der Benutzer-Schnittstelle
- Entwurf einer modularisierten Softwarestruktur
- Implementation und Test der Software hinsichtlich Funktion und Anwendertauglichkeit
- Dokumentation aller Arbeitsschritte

Daten, Anzustrebendes:

- Eingabe	Konsole
- Ausgabe	Konsole und Drucker
- Verwaltung	Datenbank
- Erweiterbarkeit	Die Anwendung soll so konzipiert werden, dass sie im nachhinein einfach um eine grafische Bedienoberfläche erweiterbar ist.

3 Projektplanung

3.1 Terminplanung

Die folgende Darstellung zeigt die detaillierte Terminplanung der einzelnen Schritte des Projekts.

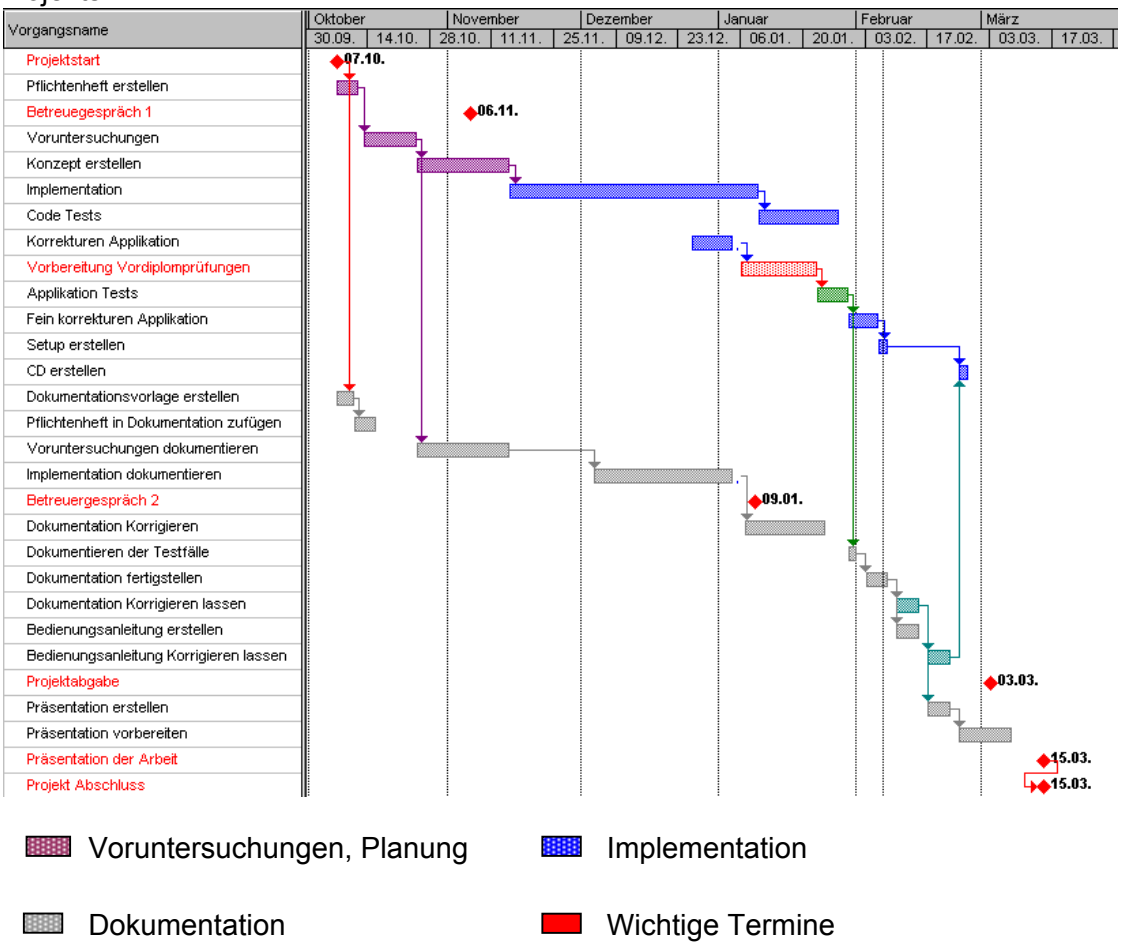

Abbildung 1: Terminplanung Projekt Flugplaner

3.2 Soll-Ist-Vergleich

Der folgende Abschnitt zeigt den Vergleich zwischen dem geplanten Arbeitsaufwand in Stunden pro Projektphase und den effektiven aufgewendeten Arbeitsstunden.

3.2.1 Voruntersuchungen, Planung

Diese Phase beinhaltet folgende Punkte:

- Pflichtenheft erstellen
- Voruntersuchungen
- Konzept erstellen

Soll: 40 Stunden
Ist: 38 Stunden

Bemerkungen zum Arbeitsaufwand/ Vorgehensweise:

- Die Einarbeitung war nicht so aufwendig wie geplant
- Für das Erstellen des Pflichtenhefts, wurde die Aufgabenstellung und das Projekteingabeformular zur Hilfe genommen
- Die Konzepterstellung hat ca. 2/3 der Arbeit ausgemacht. Den Grund dafür sehe ich darin, dass ich noch nie ein Konzept erstellt habe, und mir nicht bewusst war wieviel Aufwand es generieren würde.

3.2.2 Implementation

In der Implementation sind inbegriffen:

- Implementieren der Applikation
- Code austesten (laufend)
- Korrektur der Applikation
- Feinkorrekturen der Applikation

Soll: 50 Stunden
Ist: 80 Stunden

Bemerkungen:

- Programmierkenntnisse überschätzt
- Leerlauf durch nicht Einhalten der Projektplanungsphasen
- Aufwand für eingereichte Arbeit unterschätzt

3.2.3 Applikationen Test

Die Aufstellung beinhaltet nur die Schlusstests der fertigen Applikation. Die Code Tests wurden während der Implementation durchgeführt.

Soll: 3 Stunden
Ist: 3 Stunden

Bemerkungen:

- Da ich laufend Tests während der Implementation Phase gemacht habe, waren es eher Verbesserungsvorschläge die ich gefunden habe und keine kapitalen Fehler. Darum habe ich dafür auch nicht mehr Zeit als geplant investiert.

3.2.4 Dokumentation

Die Dokumentation beinhaltet folgendes:

- Dokumentation erstellen und korrigieren
- Betreuergespräche
- Bedienungsanleitung erstellen und korrigieren

Die Erstellung und –Vorbereitung der Präsentation ist nicht in der Dokumentation enthalten, da diese erst am Ende des Projekts gemacht wird.

Soll: 40 Stunden
Ist: 50 Stunden

Bemerkungen:

Das grosse Problem lag darin, dass mir nicht klar war wie eine Dokumentation aufgebaut werden muss und was genau darin enthalten sein muss. Dank des Betreuers konnte ich dieses Problem beseitigen. Ich habe den Fehler gemacht nicht von Anfang an eine saubere Dokumentation zu erstellen, sondern jeden Schritt nur auf Handzettel oder einzelnen Word Dokumente festzuhalten. Dadurch wurde das Zusammenstellen viel aufwändiger.

3.2.5 Unvorhergesehenes

Als Unvorgesehenes hatte ich folgende Punkte eingeplant:

- Beheben von schwerwiegenden Fehlern in der Applikation
- Leerläufe (Code mehrmals schreiben)

Soll: 10 Stunden
Ist: 20 Stunden

Bemerkungen:

Die Überschreitung der Soll-Stunden liegt in den vielen Leerläufen, welche ich vorgängig nicht eingeplant hatte (Implementation und Dokumentation). Bei gewissen für mich nicht lösbaren Problemen, musste ich Entscheidungen treffen die noch in dieser Dokumentation beschrieben werden.

3.2.6 Gesamtvergleich

Abbildung 2 zeigt den Vergleich aller geforderten und real aufgewendeten Arbeitsstunden.

Vorgang	**Soll**	**Ist**
Voruntersuchungen	40 h	38 h
Implementation	50 h	80 h
Tests	3 h	3 h
Dokumentation	40 h	50 h
Unvorhergesehenes	10 h	20 h
Total	**158 h**	**191 h**

Abbildung 2: Gesamtvergleich des Arbeitsaufwandes

Bemerkungen:

Während des ganzen Projektes habe ich den Arbeitsaufwand erfasst. Für die Erstellung der Vordiplomarbeit habe ich einen Mehraufwand von mehr als 20 % gehabt. Folgende Hauptgründe sehe ich dafür:

- Nichteinhalten der Projektphasen
- Unklarheiten bei der Erstellung der Dokumentation

Ich habe den Fehler gemacht, sofort mit der Implementation zu beginnen. Dadurch wurde das Ganze so unübersichtlich und verwirrend, dass das was ich bis anhin gemacht hatte, nicht benötigt werden konnte. Deshalb habe ich meinen Code verworfen um diesmal das Projekt nach Planung durchzuführen. Ich habe mir dann viel Zeit für die Konzepterstellung genommen und alle möglichen Varianten durchgespielt. Die für mich beste Lösung habe ich dann niedergeschrieben. Eine Implementation war dann wesentlich einfacher und übersichtlicher. Ein weiteres Problem war das Erstellen der Dokumentation. Diese Erfahrung hat mir gezeigt, wie wichtig eine saubere Konzepterstellung ist, was mir in zukünftigen Projekten sicherlich hilfreich sein wird.

4 Anforderungsspezifikation

In diesem Kapitel wird näher auf die Anforderungen der Applikation eingegangen.

4.1 Flugzeugtyp Datenbank

Für die Flugplanung und vor allem für die Schwerpunkt-Berechnung (Weight & Balance) sind diese Daten unumgänglich. Da aber immer ein Flugzeughandbuch zur Hand sein muss, um diese Daten auszulesen, wäre es von Vorteil, dass eine kleine Datenbank mit ca. 5 – 10 Flugzeugtypen in der Applikation integriert würde. Diese soll folgende Möglichkeiten haben:

4.1.1 Flugzeugtyp erfassen

Die Flugzeugtyp-Daten müssen in den in der Schweiz üblichen Masseinheiten (Liter, Kilogram und Meter) erfasst werden können. Nach dem Erfassen soll es möglich sein ca. 5 bis 10 Flugzeugtypen in eine Datenbank abzuspeichern. Auf die Daten kann dann bei der Flugplanung sehr einfach zugegriffen werden.

4.1.2 Flugzeugtyp bearbeiten

Es soll möglich sein Datenbankeinträge zu bearbeiten.

4.1.3 Flugzeugtyp löschen

Falls ein Flugzeugtyp nicht mehr verwendet wird, soll dieser wieder aus der Datenbank entfernt werden können.

4.2 Flugplanung

Die Flugplanung ist lediglich dazu da, dem Piloten einen Leitfaden für die Flugplanung zu geben. Die Planung soll verhindern dass er etwas vergisst. Die Applikation nimmt Ihm jedoch nicht die ganze Arbeit ab und übernimmt keine Verantwortung für die sichere Flugplanung. Am Ende soll ein Formular (analog zu einem Navigations-Flugplan) ausgedruckt werden können. Eine Speicherung der Flugpläne wird nicht gefordert, da jeder Flugplan für sich einzigartig ist (Gewichte, Flugzeugtyp und Route).

4.2.1 Daten erfassen

Es soll eine Erfassung des Flugzeugtyps (Datenbank), der sich an Bord befindlichen Personen (Gewichte) und der Flugroute inkl. Alternate (Ausweichflugplatz) möglich sein.

4.2.2 Berechnungen

Während der Routenplanung wird im Zusammenhang mit dem Ground Speed (GS) und der Distanz, die Flugzeit (EET) von Punkt zu Punkt berechnet. Am Schluss der Flugplanung wird ausserdem die Gesamtflugzeit sowie der Benzinverbrauch und die Weight & Balance Berechnungen ausgeführt und angezeigt. Der Pilot kann dann die Daten mit dem Flughandbuch überprüfen und entscheiden, ob ein Flug so durchgeführt werden kann.

4.2.3 Daten Ausgeben

Für jede Flugplanung muss ein Formular erzeugt werden können, welches für die Flugdurchführung benötigt werden kann. Die Darstellung ist entsprechend der, in der Fliegerei üblichen Formularen anzupassen.

4.3 Flugbuch

Das Flugbuch dient dem Erfassen der geflogenen Flugstunden. Diese sind nötig um Lizenzerneuerungen zu beantragen.

4.3.1 Daten erfassen

Folgende Daten müssen erfasst werden können: Datum, Mustertyp, Kennzeichen, Flugart, Von, Nach, Startzeit, Landezeit, Anzahl Landungen, Fluglehrer und Bemerkungen. Der Datensatz muss in Abhängigkeit des Datums an die richtige Stelle in der Datenbank eingefügt werden. Falls mehrere Einträge mit dem gleichen Datum generiert werden, wird der jeweilige Eintrag immer am Schluss der Kette eingefügt.

4.3.2 Berechnungen bei Datenerfassung

Folgende Berechnungen müssen automatisch durchgeführt und entsprechend abgelegt werden: PIC, COPI oder SIM (Kommandant, Kopiloten, Simulator) für den jeweils eingegebenen Flug.

4.3.3 Daten löschen

Das Löschen einzelner Datensätze muss möglich sein (Die Datenbank muss die entstandene Lücke automatisch schliessen können).

4.3.4 Daten Filtern

Es muss die Möglichkeit bestehen das Flugbuch in Zeitperioden zu filtern. Das heisst, dass der Benutzer die Daten für eine bestimmte Periode heraus filtern können muss. Diese Filtermethode wird benötigt, um alle zwei Jahre eine einfache Möglichkeit zu haben, dem BAZL (Bundesamt für Zivilluftfahrt) den Flugstundennachweis abzugeben.

4.3.5 Berechnung mit Filter

Folgende Berechnungen werden automatisch durchgeführt und dem Benutzer auf dem Monitor oder auf dem ausgedruckten Formular angezeigt: Gesamte Flugerfahrung, gesamte Flugstunden in der gefilterten Periode (PIC, COPI, SIM), Anzahl Landungen in der gefilterten Periode.

4.3.6 Berechnung ohne Filter

Folgende Berechnungen werden automatisch durchgeführt und dem Benutzer auf dem Monitor oder auf dem ausgedruckte Formular angezeigt: Gesamte Flugerfahrung, gesamte Flugstunden (PIC, COPI, SIM), Anzahl Landungen.

4.3.7 Daten ausgeben

Die gefilterten Daten und die gesamte Flugerfahrung können auf der Konsole oder auf dem Drucker ausgegeben werden. Diese werden auf einem entsprechenden Formular ausgedruckt das dem BAZL beigelegt werden kann.

4.3.8 Datenbank

Die Daten werden in einer Datenbank abgespeichert.

4.4 Systemdesign

Die Applikation wird in Visual C++ 6 entwickelt. Die Benutzeroberfläche wird eine Konsole sein.

4.5 Einsatz

Der Anwendungsbereich der Applikation ist die private „Hobby-Fliegerei“. Sie soll dem Piloten die Flugplanung erleichtern und ihm einen Teil der Arbeit abnehmen. Dazu wird die Qualität der Flugplanung gesteigert.

4.6 Umgebung

4.6.1 Software

Die Applikation wurde auf WINDOWS 2000 Prof. Service Pack. 3 erstellt. Die Test wurden ebenfalls auf WINDOWS XP Service Pack 1 durchgeführt. Die Software funktioniert nicht auf WINDOWS 98. Auf WINDOWS NT wurde sie nicht getestet.

Die Datei *Comdlg32.ocx* muss im System32 Ordner vorhanden sein (wird mit der Software installiert)!

4.6.2 Hardware

Die Applikation wurde auf folgender Hardware erstellt und getestet:

Prozessor:	Intel 600MHz
Speicher (RAM):	128 MB
Bildschirmauflösung:	1024 x 768

4.7 Qualitätsziele

- Saubere Dokumentation der Arbeit
- Benutzerfreundlichkeit des Tools
- Objektorientiertes Design

4.8 Test und Abnahmebedingung

4.8.1 Abnahmebedingung

Als primäre Grundlage dient die Aufgabenstellung. Weitere Anforderungen sind in diesem Pflichtenheft spezifiziert.

4.8.2 Tests

Die Tests werden anhand des Pflichtenhefts und in Bezug auf Benutzerfreundlichkeit ausgeführt. Die Test werden protokolliert und der Dokumentation beigelegt.

5 Voruntersuchungen

5.1 SW-Grundstruktur

In der folgenden Grafik ist die Grundstruktur der Applikation definiert. Die Darstellung soll die Aufgabenstellung und das Pflichtenheft grafisch darstellen.

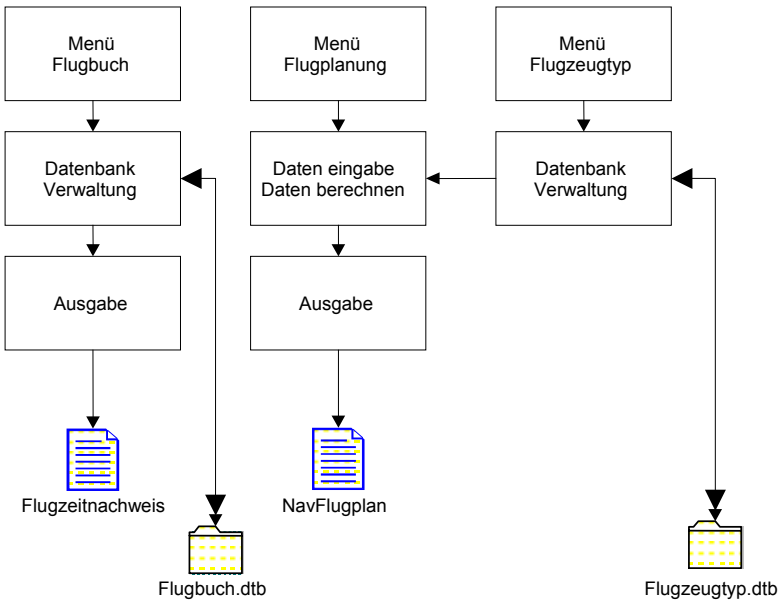

Abbildung 3: Software Grundstruktur

5.2 Datenbank

Für die Flugplanung wie auch für die Flugzeugtypen wird je eine Datenbank benötigt. Es stehen folgende Möglichkeiten zur Verfügung:

- Access Datenbank
- Excel Datenbank (*.csv Format)
- Text Datei

Auf den Zugriff einer Access Datenbank wurde aus Komplexitäts-Gründen verzichtet. Zudem genügt für diese Anwendung eine Excel- oder Textdatei.
Der Entscheid zwischen den letzten Beiden wird an diesem Punkt der Dokumentation noch offen gelassen. Der Zugriff auf die Datei wird auf jeden Fall so gestaltet, dass sowohl auf ein Text- als auch auf eine Excel-Datei zugegriffen werden kann. Für diesen Zweck müssen die Datensätze mit einem „*Komma*“ getrennt werden.

5.3 Datenausgabe

Bei einer Recherche bezüglich der für die Datenausgabe möglichen Varianten, sind folgende Möglichkeiten herausgekommen:

- Direkte Druckeransteuerung
- Visual Basic Lösung
- HTML Ausgabe

Das Problem einer direkten Drucker-Ansteuerung ist jedoch, dass die Möglichkeit sehr viele Drucker anzusteuern gegeben sein muss. Diese Variante scheint deshalb zu komplex, weil Standard Windows-Programme auf den installierten Drucker zugreifen könnten.
Deshalb wurden in der Folge einige Versuche gestartet, um die beiden anderen Varianten zu vergleichen. Basis der beiden Varianten sind, aus einer oder mehreren Dateien die Druckdaten zu lesen und diese auf ein Formular zu bringen. Diese Test-Variante wurde gewählt, damit für beide Möglichkeiten die gleichen Voraussetzungen vorhanden sind. Es bleiben damit die Möglichkeiten offen, sowohl die Visual Basic Lösung als auch die HTML Ausgabe zu implementieren.

5.3.1 HTML Ausgabe

Bei der HTML Ausgabe ist die Möglichkeit gegeben die erforderlichen Daten aus der vorbereiteten Datei auszulesen und diese dann in einen vorbereiteten HTML Code einzufügen. Danach muss das HTML File nur noch abgespeichert werden. Mit einer Methode (*shellexecute()*) kann dann die Datei mit dem Internet Explorer geöffnet werden und das Formular kann ausgedruckt werden. Aufgrund der nicht ausreichenden Kenntnisse der HTML Sprache, würde der HTML Code mit einem HTML Editor erstellt, um den Code dann in die Applikation zu integrieren.

5.3.2 Visual Basic Ausgabe

Bei der Erstellung der Visual Basic Projektarbeit „Alpaka“ musste ebenfalls ein Formular erzeugt werden können. Das Druckformular wurde mit einer Form erzeugt und diesem einen weissen Hintergrund gegeben. Die vorbereiteten Visual Basic Komponenten, geben dann die Möglichkeit, diese Form in eine Art „*Print Screen*“ auszudrucken. Für die jetzige Variante brauchen nur die erstellten Dateien

ausgelesen und die Daten in das Formular eingefügt zu werden. Mit der Druckfunktion kann dann das erstellte Formular ausgedruckt werden.

5.3.3 Entscheidung

Vorteile HTML:

- HTML ist ein sehr weit verbreiteter Code (sogar auf MAC)
- HTML Dateien können relativ einfach gestaltet werden (HTML Editor)
- HTML kann sehr einfach in den C++ Code integriert werden
- Der Internet Explorer bietet mir die Druckerauswahl

Nachteile HTML:

- Abhängig von einem Windows Programm (Internet Explorer)
- Formular Gestaltung relativ komplex
- Keine Erfahrungen

Vorteile Visual Basic Applikation:

- Erfahrungen durch Visual Basic Projekt
- Einfache Gestaltung der Formulare
- Druckmöglichkeit über die bestehenden Komponente
- Professionelles Erscheinen
- Unabhängig von anderen Programmen

Nachteil:

- COMDLG32.ocx muss auf dem Rechner installiert werden (läuft sonst nicht)

Entscheid:

Da eine unabhängige Lösung sehr wichtig erscheint und das Visual Basic Projekt ein grosser Erfolg war, fällt der Entscheid auf die Visual Basic 6.0 Lösung. Das Installieren der fehlenden OCX Datei wird über das Setup Programm realisiert.

5.4 Menü Führung

Zum Zweck der Menü-Führung wird eine Konsolen-Klasse verwendet, mit der folgende Funktionen einfach realisiert werden können:

- Text Farben setzen
- Cursor Position setzen
- Bildschirm löschen
- Konsolengrösse festlegen

Für die Menüführung im Allgemeinen wird ein immer gleich bleibender Hintergrund für die Applikation erstellt. Damit für die Applikation keine Nummern oder Zahlen eingegeben werden müssen, wird die Menünavigation über Pfeiltasten realisiert. Es soll zudem eine einfache Hilfe kontextbezogen aufgerufen werden können.
Das heisst, dass der Benutzer im Hauptmenü wie auch im Flugplanungs- sowie im Flugbuchmenü eine Taste drücken kann, die ihm dann eine grobe Hilfe bietet.

5.5 Format und Ablage der User Daten

5.5.1 Dateiformat

Die Entscheidung fiel zugunsten der Textdatei aus (vgl. Kap 5.2). Der Grund dafür ist, das beim Excel Format (*.csv) die Zahlen verändert werden (vorgängige 0 werden abgeschnitten). Beim einlesen der Datensätze aus den Dateien wird der Datensatz nicht mehr identisch sein. Um ein einheitliches Datenformat zu garantieren, wird für alle Benutzer-Dateien das gleiche Format benützt.

Die Benutzer Textdateien werden wie folgt benannt:

- *.dtb für die Datenbanken
- *.dta für Parameter Dateien
- *.txt für Hilfe-Dateien

5.5.2 Ablage Übersicht der Applikation

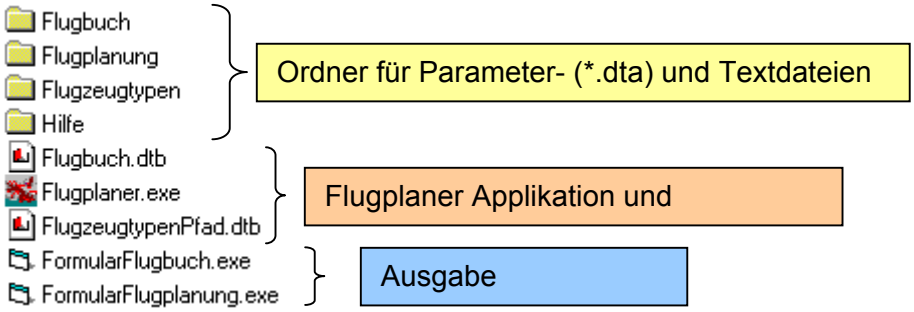

Abbildung 4: Ablage Struktur der Applikation

5.6 Installation der Applikation

Es sind für die Flugplaner-Applikation mehrere Dateiablagen vorgesehen. Zusätzlich wäre es von Vorteil, wenn der Benutzer eine Setup-Installation machen könnte. So muss sich dieser nicht sorgen wo welche Ordner erzeugt werden müssen. Zu diesem Zweck wird ein Setup Programm von Borland (Express) verwendet, welches diesen Anforderungen genügt.

6 Konzept

6.1 Menüführung

Um bei der Implementation der Applikation Zeit sparen zu können, wurde die Menüführung in der Grafik möglichst genau definiert.

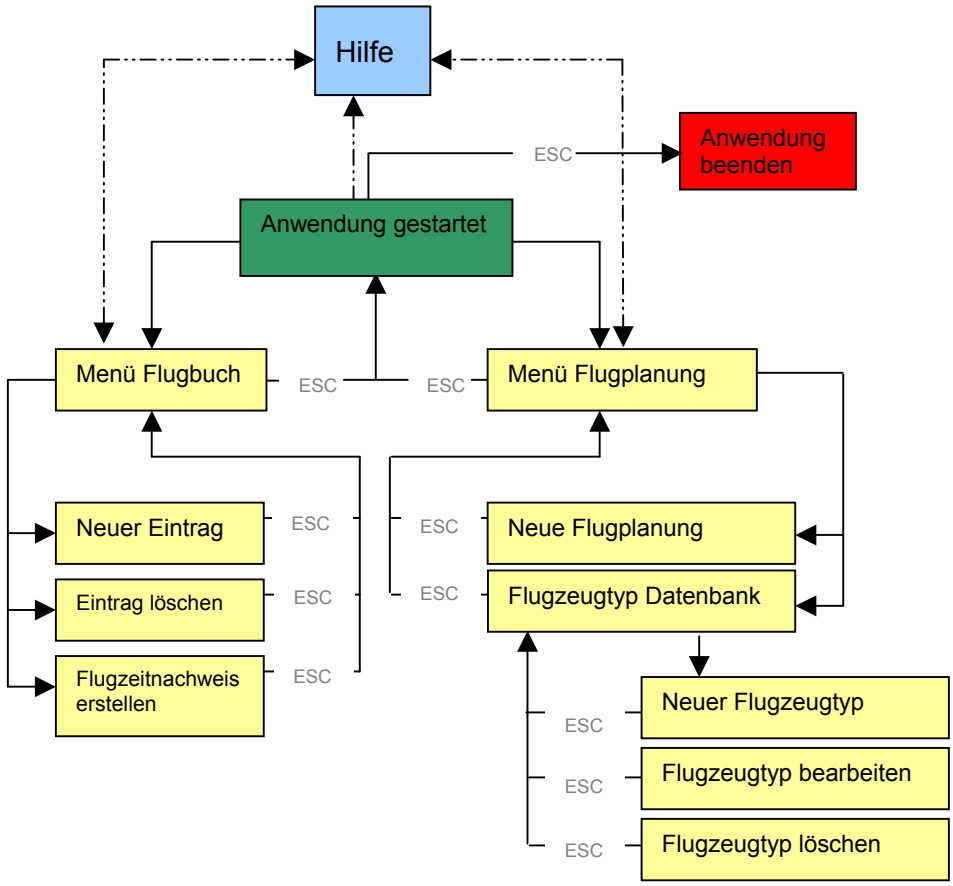

Abbildung 5: Menüführung

6.2 Hintergrund

Mangels Erfolg beim Finden einer guten Lösung für die Hintergrundgestaltung (Recherchen im Internet und eigene Programmiervergleiche), rückte das „zeichnen lassen“ des Hintergrunds in die engste Wahl. Dies bedeutet, dass bei jedem Aufrufen des Programms, der Hintergrund gezeichnet werden muss. Es wird dabei oben links begonnen und Zeile für Zeile des selbsts entworfenen Hintergrunds auf die Konsole gezeichnet.

Nachteile:

- Relativ aufwendige Lösung
- Verlangsamt die Applikation
- Bei langsamen PC’s wird der Benutzer beim Navigieren in den Menüs gebremst

Vorteile:

- Applikation macht einen guten Eindruck
- PC’s sind heutzutage so schnell, dass es niemand merken wird

6.3 Konzept Übersicht

Auf dem folgenden Bild ist die Übersicht der kompletten Applikation ersichtlich. Die Felder stellen Klassen dar, die realisiert werden sollen. Jede der Klasse (*.h) wird eine Datei (*.cpp) mit dem gleichen Namen beinhalten, die dessen Methoden enthält.

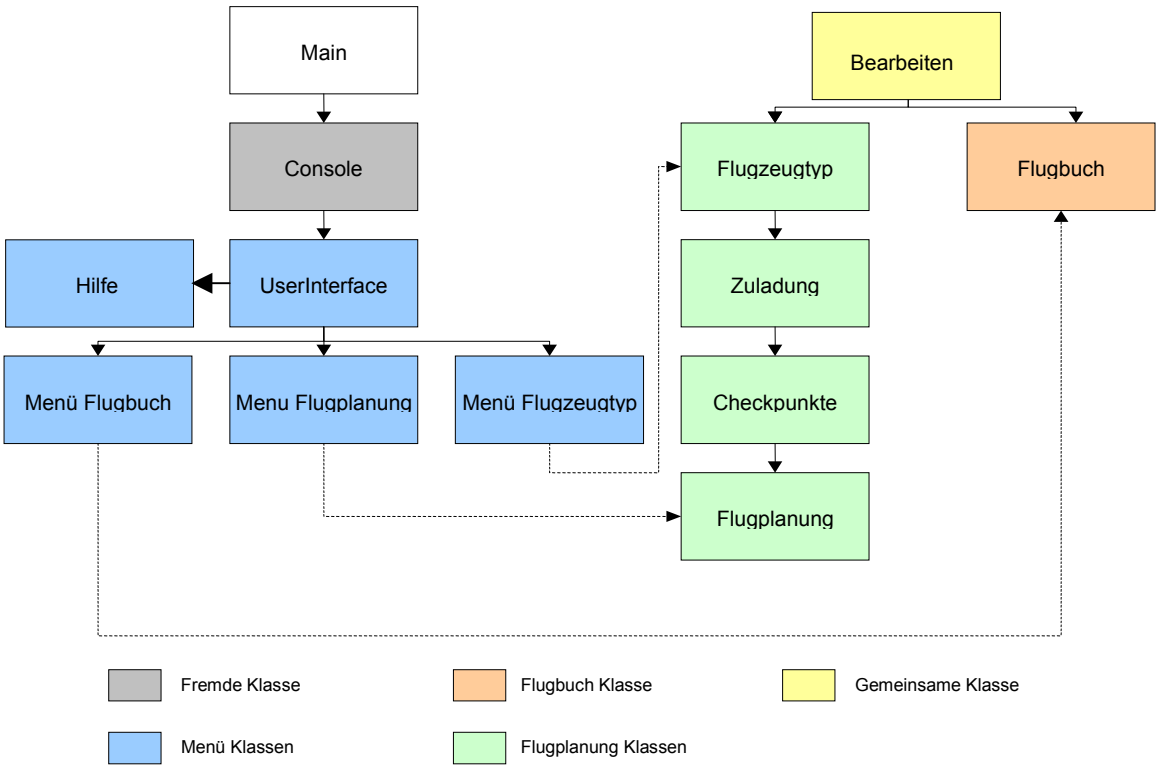

Abbildung 6: Konzept Applikation

Die Abbildung ist so zu verstehen, dass die durchgezogenen Pfeile einen direkten Zusammenhang der Klassen darstellen. Das heisst, dass die Klassen zusammen vererbt werden. Der Grund dafür ist, dass jede der untergeordneten Klassen gewisse Methoden der vorgängigen (Basis) übernehmen muss. Dies gilt für die Anwender Klassen (auf der linken Seite) sowie die Kernklassen (auf der rechten Seite). Die gestrichelten Pfeile stellen einen Zugriff dar. D.h., dass die entsprechende Klasse auf die andere zugreift. Die Schnittstelle zwischen Anwender- und Kernklassen wird so gelöst.

6.4 Konzept GUI

In diesem Abschnitt werden die Anwenderklassen beschrieben. Diese sind dazu da, die Konsolen-Oberfläche darzustellen und die Schnittstelle zu den Kernklassen zu bilden. Dieser Block könnte ebenso durch eine grafische Oberfläche ersetzt werden.

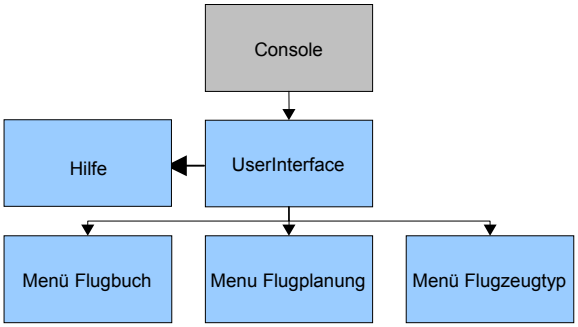

Abbildung 7: Konzept GUI

6.4.1 Console

Die Konsolenklasse stellt folgende grundsätzlichen Methoden zur Verfügung:

- Cursor Position festlegen → *locate (int zeile, int spalte)*
- Konsole leeren → *clear()*
- Farben setzen → *setColor(Text, Hintergrund)*
- Konsolen Titel setzen → *setTitel(const::string&title)*
- Grösse der Konsole festlegen →*setSize(int width, int height)*

6.4.2 User Interface

Der User Interface wird der Hauptbestandteil dieses Blockes. Dieser beinhaltet alle gemeinsamen Methoden der Anwenderklassen:

- Darstellung des Hintergrundes
- Menüpunkte zufügen/ anzeigen
- Bild löschen
- Cursor setzen (Pfeil)
- Setzen der gemeinsamen Eingabe Felder/ Task-Menüs

6.4.3 Menü Flugbuch

Die Menü-Flugbuch Klasse wird dem Benutzer die Menüführung für das Verwalten der Flugbuch Datenbank liefern. Ausserdem bildet sie die Schnittstelle zum Kern der Applikation wo die Methoden der Flugbuchklasse aufgerufen werden.

- Menü für das Flugbuch setzen (neue Eingabe, Eintrag löschen, Flugzeitnachweis erstellen)
- Daten Eingabe (Eingabe der Flugbuch Daten über die Konsole)
- Daten filtern (zeigt Flugzeitnachweis Menü auf der Konsole an und ruft Filterfunktion des Flugbuches auf)
- Daten löschen (zeigt Lösch Menü auf der Konsole an und ruft Löschmethode der Flugbuchklasse)
- Erstellen der Flugzeitnachweis Datei

6.4.4 Menü Flugplanung

Die Menü-Flugplanung Klasse wird dem Benutzer die Menüführung für die Flugplanung liefern. Auch diese bildet die Schnittstelle zum Kern der Applikation, wo die Methoden der Flugplanung und dessen Basisisklassen aufgerufen werden.

- Menü für die Flugplanung setzen (neue Flugplanung, Flugzeugdatenbank)
- Auswahlmenü der Flugzeugtypen (zeigt die vorhandenen Flugzeugtypen an, der Benutzer kann einen auswählen)
- Gewicht Erfassungsmenü anzeigen und Datenelemente der Zuladung-Klasse setzen
- Flugrouten-Menü anzeigen und Datenelemente des Startflugplatzes setzen
- Datenelemente der Checkpunkte Klasse setzen
- Zuladung berechnen (ruft Berechnungsfunktion der Zuladung und übergibt die nötigen Elemente)
- Moment berechnen (ruft die Momentberechnung auf und übergibt die nötigen Elemente)
- Berechnung Flugzeiten (berechnet die Flugzeiten EET und zeigt sie auf der Konsole an)

- Flugroute speichern (ruft alle Speichermethoden der gesamten Flugroute auf [Startfulgplatz.dta, Route.dta, Zuladung.dta, Alternate.dta, Berechnungen.dta])

6.4.5 Hilfe

Die Hilfeklasse wird für die Benutzerhilfe erstellt. Diese kann im Haupt- sowie in den Untermenüs (Flugbuch und Flugplanung) aufgerufen werden können. Es werden drei Textdateien erzeugt, bei welchen die Hilfe beschrieben wird. Die Applikation wählt diese Dateien aus, um sie auf der Konsole darzustellen.
Der Benutzer wird dann die Möglichkeit haben mit Pfeiltasten die Dateien zu navigieren. Die Methoden, welche die Klasse beinhalten sollte, sehen wie folgt aus:

- Hilfemenü anzeigen
- Hilfefiles aufrufen (Allgemein-, Flugbuch- und Flugplanungshilfe)
- Hilfefiles auf der Konsole anzeigen
- Navigieren in den Hilfefiles

6.4.6 Menü Flugzeugtyp

Das Menü Flugzeugtyp-Klasse wird dem Benutzer die Menüführung für das Verwalten der Flugzeugtyp Datenbank liefern. Auch diese bildet die Schnittstelle zum Kern der Applikation, wo die Methoden der Flugzeugtypklasse aufgerufen werden.

- Setzen des Flugzeugtyp Menü (neuer Flugzeugtyp, Flugzeugtyp bearbeiten, Flugzeugtyp löschen)
- Setzen „Neuer Flugzeugtyp-Menü“ (zeigt Eingabemenü an und setzt nach der Benutzereingabe die Datenelemente der Flugzeugtyp-Klasse)
- Setzen „Flugzeugtyp bearbeiten“ Menü (ermöglicht die Auswahl des zu bearbeitenden Flugzeugtyps und ermöglicht die Bearbeitung der verschiedenen Parameter)
- Setze „Flugzeugtyp löschen-Menü“ (ermöglicht die Auswahl des zu löschenden Flugzeugtyps, danach entfernt es die Datei und den Datenbankeintrag aus der FlugzeugtypPfad.dtb)

6.5 Konzept Flugplanung

In diesem Abschnitt wird ein Teil der Kern-Klassen beschrieben. Diese beinhalten die Hauptfunktionen der Applikation in diesem Fall der Flugplanung. An diesen Blöcken könnte auch eine grafische Bedienoberfläche angeschlossen werden.

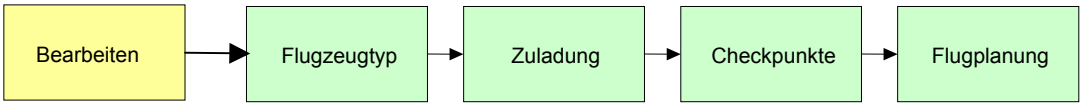

Abbildung 8: Konzept Flugplanung

6.5.1 Flugzeugtyp

Es besteht der Wunsch, einerseits eine Datenbank zu erstellen, die alle (max. 10) Dateinamen enthält und anderseits in einem speziell dafür geschaffenen Ordner die entsprechenden Dateien mit den Flugzeugtypen-Daten anzeigt. Das bedingt jedoch, dass beim Erzeugen der Dateien ein Dateinamen definiert wird, mit dem die Datei abgespeichert werden kann. Der Dateiname wird dann in die Datenbank eingetragen, so dass der Benutzer weiss, welche Dateien ihm zu Verfügung stehen. So hat der Benutzer die Möglichkeit, die richtige Datei zu öffnen und zu bearbeiten. Beim Löschen eines Eintrages sollte der Datenbankeintrag (Dateiname) sowie die entsprechende Datei gelöscht werden können.

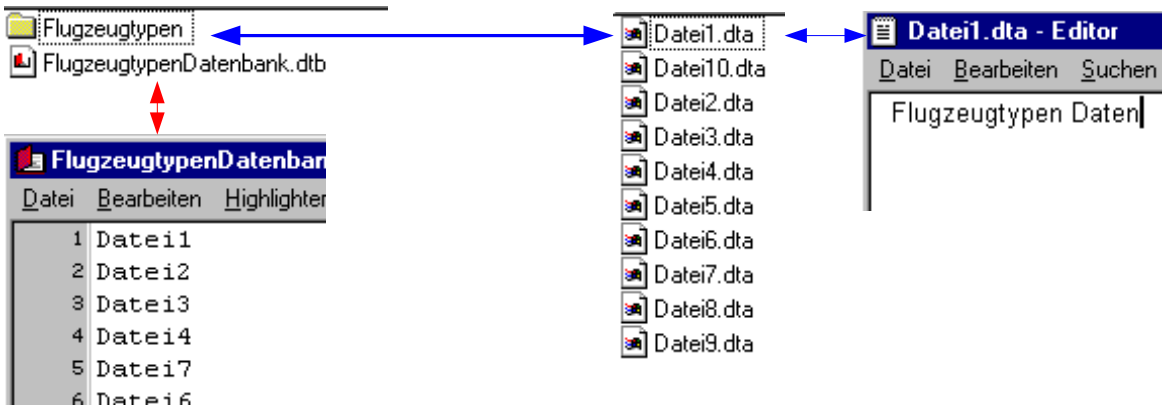

Abbildung 9: Konzept Speicherung Flugzeugtyp

Die Flugzeugtyp Klasse ist die Basisklasse der Zuladung-Klasse und soll folgende Methoden enthalten:

- Holen und setzen der privaten Datenelemente
- Setzen des Filenamen (Flugzeugtyp)
- Schreiben und lesen des Filenamen in der Flugzeugtyp-Datenbank
- Schreiben und lesen der Flugzeugtyp-Daten in den definierten Dateien
- Löschen der Datei und des Eintrages in der Datenbank

6.5.2 Bearbeiten

Die Klasse „Bearbeiten“ ist die Basisklasse der Flugzeugtyp-Klasse. Die Klasse wird sowohl für das Flugbuch, als auch für die Flugplanung benötigt. In dieser Klasse sollen alle gemeinsamen Methoden der Flugplanung und des Flugbuchs eingebaut werden. Es sind dies folgende Methoden:

- Wandeln von Datentypen
- Überprüfen von Datensätzen

6.5.3 Zuladung

Die Zuladungsklasse wird erstellt, um die Zuladungen die für einen Flug nötig sind zu setzen. Diese wird die übergebenen Daten in eine dafür vorgesehene Datei abspeichern. Die Daten können dann wieder aus der Datei gelesen werden, um das Formular zu erstellen. Folgende Methoden sind dafür notwendig:

- Setzen und holen der privaten Datensätze
- Schreiben der Zuladungdatei

6.5.4 Checkpunkte

Die Checkpunkteklasse wird die Klasse für die Routenerfassung. In dieser Klasse werden die Daten erfasst und in drei verschiedenen Dateien abgespeichert (Startflugplatz, Route und Ausweichflugplatz). Die Datei wird wie die Zuladungs-Datei bei jeder Planung überschrieben. Auch hier werden die Dateien wieder ausgelesen, um das Flugplanungs-Formular zu erstellen. Folgende Methoden werden darin enthalten sein:

- Erstellen eines Datenarray für die Routenplanung (ca. 15 Einträge)
- Setzen und holen aller privaten Datensätze
- Schreiben der Routendateien

6.5.5 Flugplanung

Die Flugplanung-Klasse ist für die Berechnungen der nötigen Parameter zuständig, ausserdem beinhaltet sie noch die Benzinverbrauchswerte der Flugplanung. Alle nötigen Berechnungen wie die Zeit von Punkt zu Punkt, Schwerpunkt, Gewicht und Benzinverbrauch werden hier getätigt. Zudem werden hier die Benzinverbrauchs Daten in eine Datei abgespeichert, die bei der Formularerzeugung wieder ausgelesen werden. Die Flugplanung-Klasse wird folgende Methoden beinhalten:

- Setzen und holen der privaten Datensätze
- Berechnung der Flugzeit, Schwerpunkt, Gewicht und Benzin Verbrauch
- Daten in Datei schreiben

6.6 Konzept Flugbuch

In diesem Abschnitt wird der Rest der Kern Klassen beschrieben. Die gelben Felder werden wie vorgängig schon gesehen (vgl. Kapitel 6.3) für alle Kern Klassen benötigt. Für den Flugbuch-Teil wird nur eine Klasse vorgesehen.

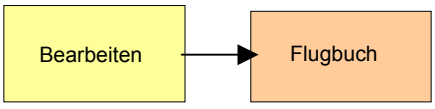

Abbildung 10: Konzept Flugbuch

6.6.1 Flugbuch

Die Flugbuchklasse wird alle Methoden die für die Datenbankverwaltung nötig sind beinhalten. Diese wird dem Benutzer die Möglichkeit geben Datensätze datums abhängig in die Datenbank einzufügen, Einträge zu löschen und Daten zu filtern um einen Flugzeitnachweis erstellen zu können. Für diese Funktionen sollen folgende Methoden implementiert werden:

- Setzen und holen der privaten Datensätze
- Daten in Datenbank schreiben und auslesen
- Daten in Datenbank (nach Datum) einfügen
- Daten aus Datenbank entfernen
- Datenfilter (Start und Stop Datum)
- Berechnung der Flugzeiten (für Flugzeitnachweis)

6.7 Konzept Visual Basic Ausgabe

Folgende Abbildung zeigt den Aufbau der Visual Basic Formular Applikationen.

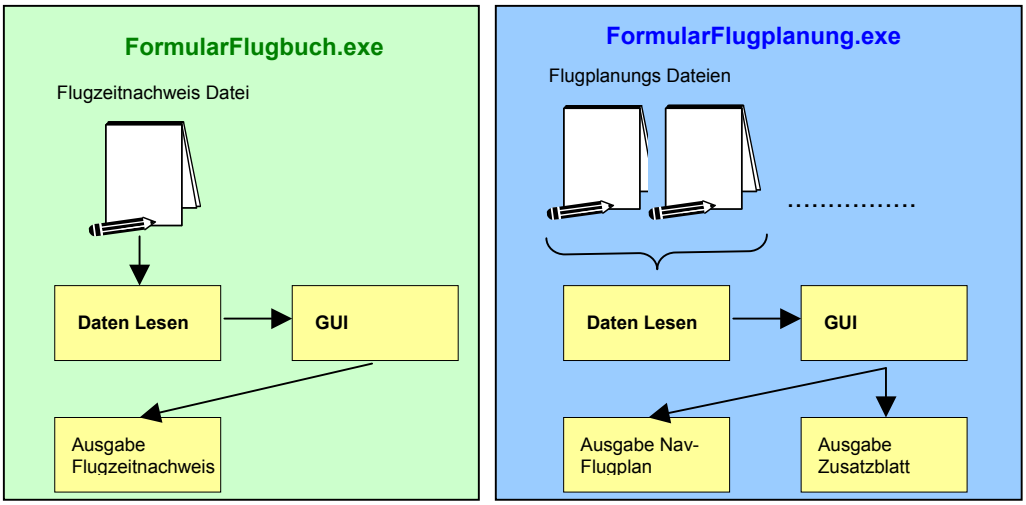

Abbildung 11: Konzept Ausgabe Module

6.7.1 Allgemein

Die Ausgabeapplikationen werden analog und gemäss Voruntersuchungen mit Visual Basic aufgebaut. Die einzigen Unterschiede werden darin bestehen, dass beim Flugbuch nur eine und bei der Flugplanung mehrerer Dateien ausgelesen werden. Ausserdem wird für das Flugbuch nur ein Formular und für die Flugplanung mehrere Formulare erstellt. Das Ganze wird so gelöst, dass die entsprechenden Datei(en) gelesen werden und in das fertige Formular eingefügt werden. Der Benutzer kann dann auswählen, ob er eine Vorschau wünscht, oder ob er das Dokument direkt ausdrucken will.

6.7.2 Daten Lesen

Die Daten werden aus der Datei gelesen um sie für die GUI (Form) zur Verfügung zu stellen.

6.7.3 GUI

Die GUI besteht aus zwei bzw. drei Teilen.

- Auswahlfenster
- Formularfenster

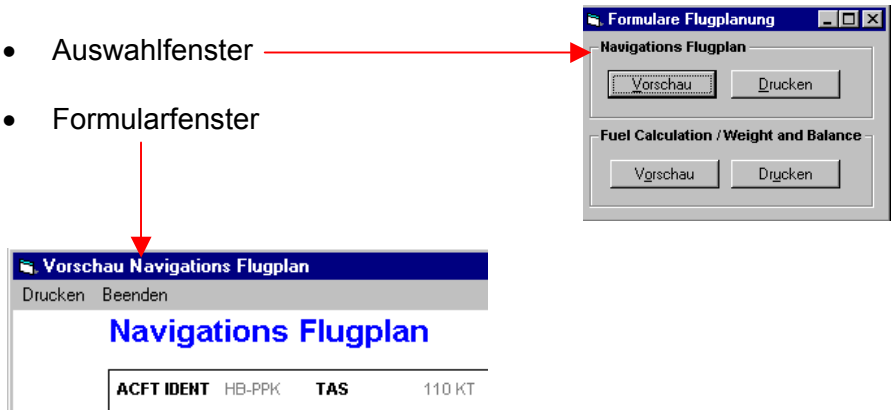

Abbildung 12: GUI der Visual Basic Applikation

Beim Auswahlfenster kann entweder direkt ausgedruckt werden oder eine Vorschau beantragt werden. Im Formularfenster (beim Flugbuch nur eine) kann das Formular angeschaut und dann ausgedruckt werden.

6.7.4 Ausgabe

Wenn der Benutzer die Druckfunktion auswählt, wird Ihm eine Druckerauswahl (Windows Standart) zur Verfügung gestellt. Darin hat er die Möglichkeit den Drucker anzuwählen, um das angezeigte Formular auszudrucken.

7 Implementation

In diesem Abschnitt wird nicht auf die einzelnen Methoden eingegangen. Die Klassen und deren Methoden wurden während der Konzept Erstellung definiert. Ausserdem ist der Code gut beschrieben.

7.1 Flugbuch Klassen

7.1.1 Probleme

Bei der Implementierung der Flugbuch-Klasse sind einige Probleme und Fragen aufgetaucht, die es zu lösen galt:

- Dynamisches Erstellen von Klassen
- Umwandlung von Datentypen (String → Int, Int → String usw.)
- Filtermethode (Wie kann ein bestimmtes Datum in der Datenbank gefunden werden, welches den entsprechenden Index zurückgeben kann. Was geschieht wenn das gesuchte Datum nicht vorhanden ist?)
- Wie kann ein Eintrag in der Datenbank gelöscht werden, damit alle Einträge nachrutschen?
- Wie kann ein Eintrag zwischen zwei Einträgen eingefügt werden?
- Daten schreiben / lesen aus der Datei

7.1.2 Hürden

Das Problem des Erstellens einer dynamischen Klasse (im Stil NEW und DELETE) kann nicht überwunden werden. Die einzige Lösung lag somit im Generieren eines fixen Datenarray von 2000 Einträgen. Erfahrungsgemäss werden 30 bis 40 Flüge pro Jahr erfasst, d.h., dass der Datenarray für 50 Jahre Flugbuch Einträge reichen würde.

7.1.3 Problemlösungen

Zu den anderen Problemen wurden folgende Lösungen erarbeitet oder gefunden:

- **Umwandlung von Datentypen**

  Die Umwandlung von Datentypen kann über den Datentyp stringstream (#include <sstream>) wie folgt realisiert werden:

```
String  Text          = „123“;
double DoubleWert = 0.0;

stringstream stream;
stream << Text;
stream >> DoubleWert;
```

Abbildung 13: Code Beispiel (www.devmentor.ch)

Die gleiche Methode kann auch für andere Datentypen verwendet werde.

- **Filtermethode**

  Für die Filtermethode wurde auf einer Seite die Datenbank mit einem Start und einem Stop Parameter versehen (00.00.0000 erster Eintrag und 99.99.9999 letzter Eintrag). Die Datums (STRING) werden in ein Zahlenformat (INT) umgeändert, danach wird nach einem korrespondierenden Jahr, Monat und Tag (zwischen dem ersten und dem letzten Eintrag) gesucht. Falls das Datum existiert, wird mir der Index zurückgegeben. Falls nicht, wird der nächsthöhere Index zurückgegeben.
  Für das Stopdatum (Bis) wird das Gleiche gemacht, jedoch in umgekehrter Reihenfolge. D.h. die Datei wird von deren Anfang bis zum Ende nach korrespondierenden Datums durchsucht. Wenn keine gefunden werden, liefert die Methode das nächsttiefere. So können die Indexe für die Flugzeitnachweis Erstellung erhalten werden.

- **Löschen eines Datenbankeintrages**

  Für das Löschen muss der Löschmethode ein Löschindex übergeben werden. Der entsprechende Datensatz mit dem angegebenen Index wird aus dem Array gelöscht und alle nachfolgenden Datensätze rutschen einen Schritt nach. Am Schluss der Methode wird der Datensatzcounter um 1 verkleinert.

- **Einfügen eines Eintrages in die Datenbank**

  Um Daten einzufügen, wird umgekehrt wie bei der Löschung von Daten vorgegangen. Der Methode muss der Einfüg-Index übergeben werden, wo ein Eintrag eingefügt werden soll (diesen erhält man von der Filtermethode). Danach muss der Counter um 1 erhöht und alle Datensätze ab dem Einfügindex in den nächsthöheren Index kopiert werden. Somit wird im Array ein Platz frei. Dort kann jetzt der neue Datensatz hingeschrieben werden. Nun müssen nur noch die Datenelement wieder in die Datenbank abgelegt werden (Daten speichern).

- **Daten in eine Datei schreiben und daraus lesen**

  Damit die Daten auch in *.csv Dateien eingefügt werden könnten, werden diese Elemente mit Kommas getrennt (Aus dem Buch „easy C++“ Kapitel 15)

7.1.4 Erkenntnisse

Bei der Konzeptionierung der Klassen, hätte genauer auf die einzelnen Funktionen eingegangen werden sollen, um Leerläufe zu vermindern.

7.2 Flugplanung Klassen

7.2.1 Probleme

Folgende Frage trat im Zusammenhang mit den Flugplanung-Klasse auf:

- Wie kann eine Datei gelöscht werden?

7.2.2 Problem Lösung

Die Löschfunktion in der Datenbank konnte ähnlich gestaltet werden wie im Flugbuch. Um eine ganze Datei zu löschen, habe ich im Internet (www.c-plusplus.de) folgende Lösung gefunden:

remove(Pfad und Datei Name)

7.2.3 Erkenntnisse

Es war wesentlich einfacher die Flugplanungs-Klassen zu implementieren, da ich aus der Erfahrung der Flugbuch Klasse sehr profitiert habe.

7.3 Bearbeiten-Klasse

Die gemeinsamen Klassen sind während der Entsehung des Flugbuchs und der Flugplanungs-Klassen entstanden. Sie beinhalten allgemeine Methoden, die beide Klassen benötigen. Dort sind auch die Erfahrungen aus dem Wandeln von Dateitypen eingeflossen.

7.3.1 Probleme

Für die Erstellung der GUI werden zusätzlich noch folgende Methoden benötigt:

- Einlesen des ASCI Code der Tastatur für die Benutzereingaben
- Prüfen von Datensätze wie Datum, Flugart und Flugzeiten
- Einlesen der Tastatur über getch()

7.3.2 Problemlösung

Die oben beschriebenen Methoden zu Implementieren war weniger ein Programmierproblem, als die Vorstellung wie es genau gelöst werden sollte. Um das Ganze etwas übersichtlicher zu machen, sollte immer die gleiche Methode verwendet werden um die Daten einzulesen. Zusätzlich muss das Eingabefeld positioniert werden können. Es gelang, eine einheitliche Methode für die Applikation zu erstellen, bei der folgende Funktionen möglich sind:

- Tastatureingaben werden angenommen
- Beim drücken von ESCAPE springt der Benutzer ein Menü zurück
- Beim drücken der Backspace-Taste kann der ganze Datensatz gelöscht werden (einzelne Zeichen zu löschen werden als zu aufwendig beachtet)
- Mit der Eingabe von ENTER oder der TAB Taste wird der eingegebene Text zurückgegeben
- Die Methode kann für die ganze Applikation verwendet werden

Das Prüfen von Datensätzen konnte wie folgt gelöst werden:

- Wandeln der STRING Datenelemente in Zahlen oder CHAR
- Überprüfen der Werte
- Rückgabe von True oder False

7.3.3 Erkenntnisse

Die Bearbeiten-Klasse ist ein Sammelsurium von Methoden die schwierig zu platzieren waren. Es ist so eine Mischklasse zwischen GUI Klassen und Kern Klassen. Eine genauere Konzeptionierung hätte eine solche Klasse sicherlich unnötig gemacht.

7.4 GUI Klassen

Aufgrund der intensiven Beschäftigung mit diesem Thema bei der Planung und der Konzeptphase des Projekts, war dessen Implementation nicht mehr sehr schwierig.

7.4.1 Probleme

Während der Implementation der GUI-Klassen sind sehr viele Probleme aufgetaucht. Das grösste Problem war, die Gestaltung der Oberfläche.

Zusätzlich waren folgende Probleme von grosser Bedeutung:

- Steuerung über Pfeiltasten
- Bearbeiten der eingegebenen Datensätze (Flugbuch, Flugzeugtyp und Flugplanung)
- Abspeichern der Flugplanungen für die Wiederverwendung (nicht gefordert)
- Löschen von Flugbucheinträge

7.4.2 Hürden

- **Abspeichern der Flugplanungen zur Wiederverwendung**

  Die vorhandene Idee konnte nicht umgesetzt werden, da nicht alle Daten wieder-verwendet werden konnten. Den es verläuft jeder Flug anders (abhängig von Gewicht, Route, Flugzeugtyp).

- **Bearbeiten der eingegebenen Datensätze (Flugbuch, Flugzeugtyp und Flugplanung)**

  Das Wandern in den einzelnen Eingabefeldern wurde durch die Konsolen-Applikation verunmöglicht. Eine grafische Oberfläche mit Mausbedienung wäre hier dienlicher gewesen.

7.4.3 Entschlüsse

- **Abspeichern der Flugplanungen zur Wiederverwendung**

  Nach beträchtlichem Zeitaufwand wurde aus folgenden Gründen entschieden das Projekt zu verwerfen:

  - In der Aufgabenstellung nicht gefordert
  - Wird in der Praxis zu 90% nicht benötigt da jeder Flug einmalig ist
  - Die Applikation wäre viel zu unübersichtlich geworden
  - Der Zeitaufwand kann besser genutzt werden
  - Könnte bei einer Erweiterung mit einer grafischen Bedienoberfläche nachgeholt werden

- **Bearbeiten der eingegebenen Datensätze (Flugbuch, Flugzeugtyp und Flugplanung)**

  Folgende Entscheide musste aufgrund des Konzepts durchgeführt werden:

  - Alle Eingaben funktionieren nach dem gleichen Prinzip. Wenn ein Datensatz eingegeben wird, kann dieser mit der Back-Space Taste wieder gelöscht werden. Wenn ein zweites Mal die Backspace-Taste gedrückt wird, springt der Cursor ein Feld zurück und löscht den entsprechenden Eintrag. So können Fehleinträge korrigiert werden.
  - Beim Flugzeugtyp besteht zusätzlich die Möglichkeit, die Eingaben nach dem Abspeichern zu bearbeiten. Dies war nötig, da es sehr gut möglich ist, dass sich die Parameter eines Flugzeugtyps verändern. So können Korrekturen gemacht werden, ohne den ganzen Datensatz verwerfen zu müssen. Für diesen Zweck muss der Benutzer den zu bearbeitenden Flugzeugtyp anwählen und dann angeben, welche Parameter er ändern möchte. Nach der Änderung muss er den Eintrag bestätigen. Dieser wird dann in der entsprechenden Datei abgespeichert.
  - Für die Flugplanung kann jeweils nach jeder Routenplanung die aktuelle sowie die vorgängige Zeile gelöscht werden. Diese Funktion ist hier nötig, weil bei der Flugplanung die Möglichkeit besteht, dass die Route ändert.
  - In jedem Fall gilt: Der Parameter der gerade eingegeben wird, kann immer mit der Backspace-Taste gelöscht werden. Mit ESCAPE kann jede Eingabe abgebrochen werden und ein Menü zurückgegangen werden.

7.4.4 Problemlösung

- **Steuerung über Pfeiltasten**

  Dieses Problem konnte über die kbhit() Funktion gelöst werden. Die Funktion wartet auf einen Tastendruck. Nun braucht nur noch der ASCII Code der Taste angegeben zu werden und es funktioniert.

- **Löschen von Flugbucheinträgen**

  Das Löschen wurde so realisiert, dass mit Pfeiltasten der Datenbankinhalt mit allen Datensätzen angeschaut werden kann und dass mit der manuellen Eingabe des zu löschenden Indexes der Eintrag aus der Datenbank entfernt wird (Löschmethode in der Flugbuch-Klasse)

7.4.5 Erkenntnisse

Sich schon zu Beginn auf das optische Erscheinungsbild zu versteifen lässt einem in Kleinigkeiten verharren und bringt einiges an Mehraufwand. Der Schwerpunkt sollte deshalb auf die Funktionalität gesetzt werden und beim Feinschliff der Software können dann die nötigen Korrekturen gemacht werden.

7.5 Aufbau der Dateien

Zur Laufzeit der Flugplanungsapplikation werden diverse Dateien für die Formular Erstellung erzeugt. Um zusätzliche Implementationen zu ermöglichen, wird dessen Inhalt im Anhang (vgl. Kapitel 9.6) beschrieben.

7.6 Visual Basic Ausgabe

Für die Ausgabe konnte ich das Projekt „Alpaka“, dass wir im Visual Basic Unterricht in einer Vierer Gruppe erstellt hatten, zu Hilfe nehmen.

7.6.1 Probleme

- Auslesen von ASCII Zeichen aus einer Textdatei
- Druckfunktion

7.6.2 Problemlösung

- **Auslesen einer Datei**

Die Lösung dieses Problems liegt im Erstellen eines Datenarray pro Datei, in welchem dann die Datensätze Zeile für Zeile eingelesen werden können. Folgendes Beispiel verdeutlicht dies:

```
FileNr = FreeFile
File = App.Path & FileName   'File öffnen

Open File For Input As FileNr
Do While Not EOF(FileNr      'Schleife bis Dateiende.
Line Input #FileNr, Array (zaehler)     'Array abfüllen
zaehler = zaehler + 1
Loop

Close #FileNr                'File schliessen
```

Abbildung 14: Code Beispiel (TSU Visual Basic Ordner)

- **Druckfunktion**

Um die Druckfunktion zu realisieren, muss in Visual Basic das *Microsoft Common Dialog 6.0* zugefügt werden. Danach konnte folgende Möglichkeit ausgenutzt werden:

```
Dim BeginPage, EndPage, NumCopies, i ', Msg, HWidth, HHeight
Dialog.CancelError = True        ' Cancel auf True setzen
On Error GoTo ErrHandler         ' Dialogfeld "Drucken" anzeigen
```

```
Dialog.ShowPrinter
BeginPage = Dialog.FromPage  ' Vom Benutzer ausgewählte Werte vom Dialogfeld abrufen
EndPage = Dialog.ToPage
NumCopies = Dialog.Copies
For i = 1 To NumCopies
      Load frmForm1              ' Hier Code zum Senden von Daten an den Drucker einfügen
     Form1.PrintForm
     Unload Form1
Next i

Exit Sub
ErrHandler:                      ' Benutzer hat Schaltfläche "Abbrechen" gewählt
```

Abbildung 15: Code Beispiel (TSU Visual Basic Ordner)

7.7 Diagramm Flugbuchformular

Auf dem folgenden Diagram ist eine grobe Übersicht der Flugbuchausgabe Applikation ersichtlich. Die einzelnen Methoden sind im Code beschrieben.

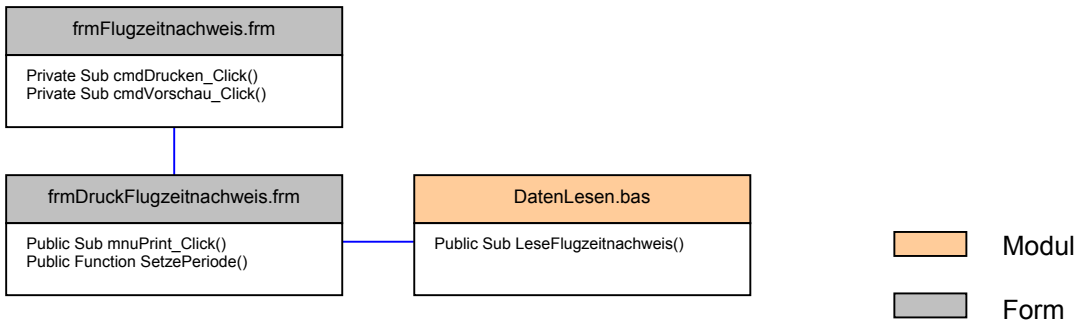

Abbildung 16: Diagramm Flugbuchformular

7.8 Diagramm Flugplanungformular

Analog zu der Flugbuchausgabeapplikation sind die Methoden im Code beschrieben.

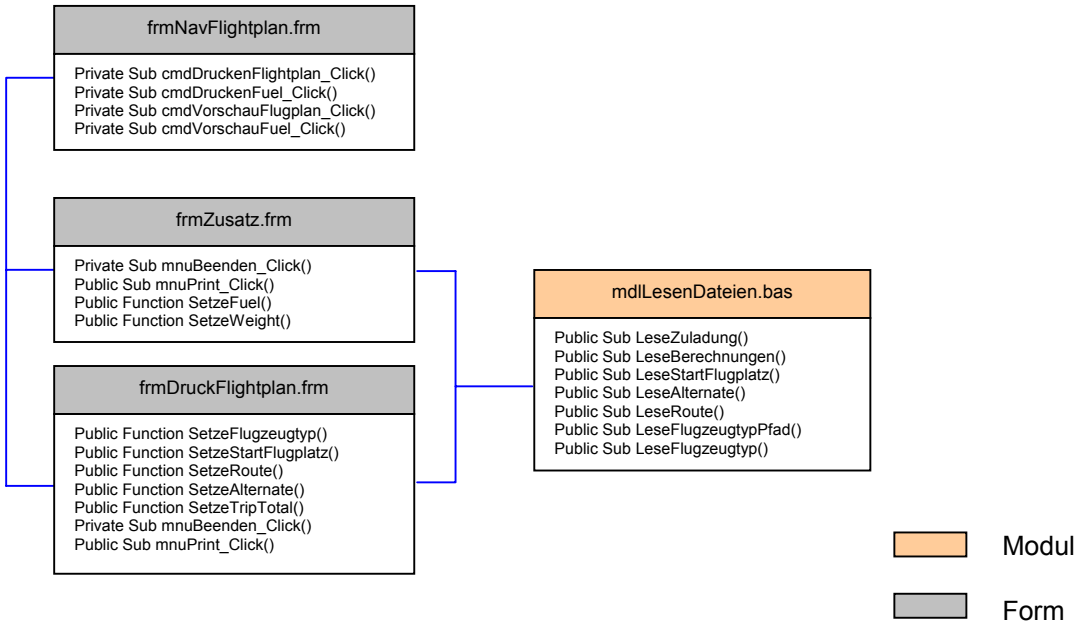

Abbildung 17: Diagramm Flugplanungformular

7.9 Klassendiagramm Flugplaner

Die Klassen Datei (*.h) beinhaltet immer eine gleichnamige Methode (*.cpp). Die Methoden sind im Code beschrieben.

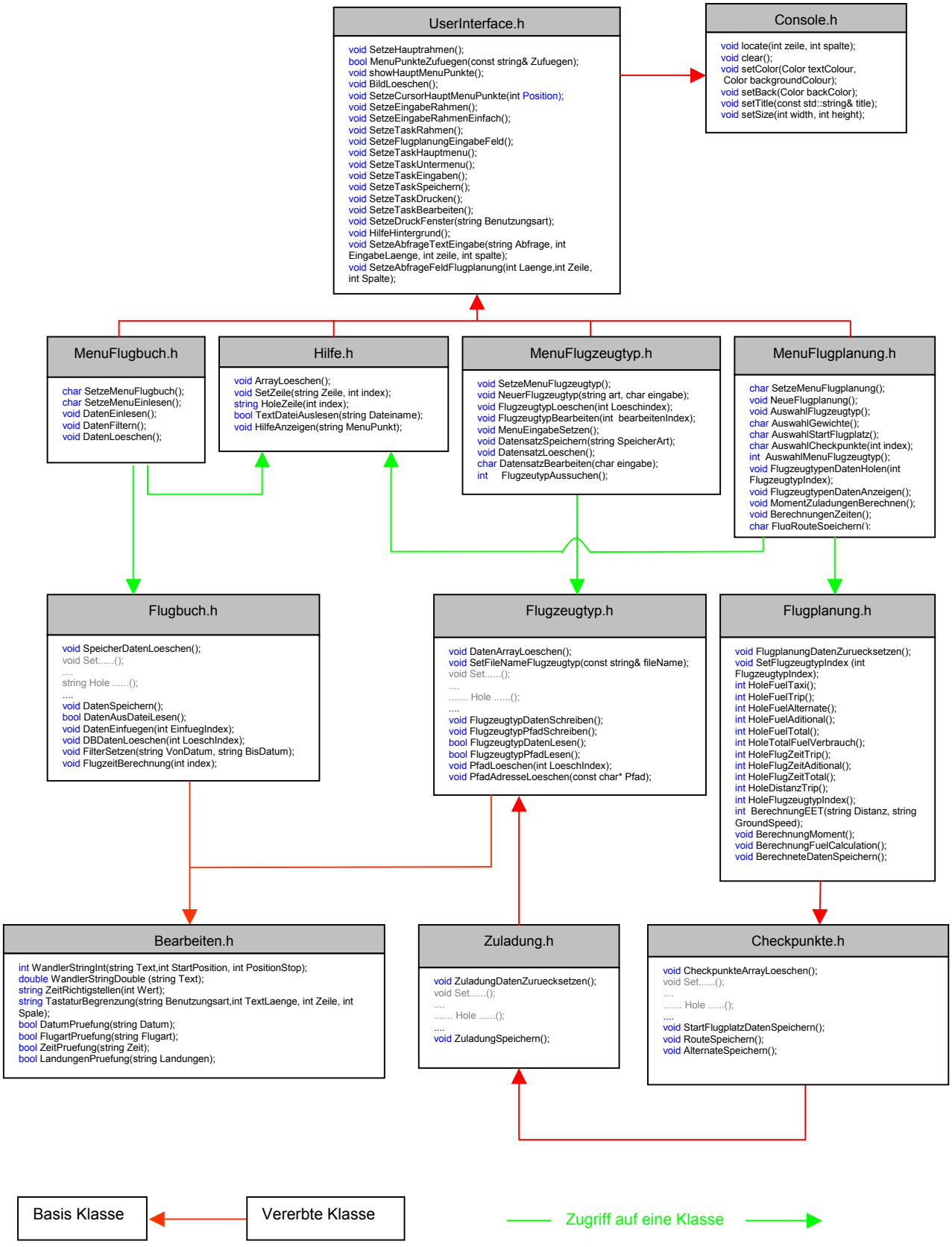

Abbildung 18: Klassendiagramm

8 Tests

Das Testen der Applikation ist neben einem gut erarbeiteten Konzept das Wichtigste am ganzen Projekt. Das Testen des Programms ist in vier Phasen abgelaufen. Die ersten Tests wurden gleich nach der Erstellung der einzelnen Methoden gemacht. In der zweiten Phase des Testens wurde das Programm als Ganzes mit Hilfe des erstellten Testdrehbuches (vgl. Kapitel 9.5) getestet. Dieses wurde dann Punkt für Punkt durchgearbeitet, um sicherzustellen, dass die Applikation auch gemäss den Anforderungen einwandfrei funktioniert. Danach wurden Benutzertests durchgeführt, um sicherzustellen, dass die Applikation benutzerfreundlich aufgebaut ist. Die Korrekturen flossen in die Applikation. Vor Projektabgabe wurde dann mit der korrigierten SW Version nochmals das Testdrehbuch (vgl. Kapitel 9.5) durchgespielt.

8.1 Testversionen

Um die Übersicht während der ganzen Testreihen zu gewähren, wurden Testversionen erstellt. Im Folgenden ist eine Tabelle mit den Versionen und den behobenen Fehlern dargestellt. Die Korrekturen die durchgeführt wurden, sind ebenfalls im Code beschrieben. Nach den Tests wurde die Version auf 1.0 gesetzt, welche diesem Dokument beiliegt (vgl. Kapitel 9.7).

Version Nr.	Testphase	Fehler	Behoben in
Testversion 0.0	Testdrehbuch	Es können mehr als 10 Flugzeugtypen erfasst werden → Programm stürzt ab!	Testversion 0.1
Testversion 0.0	Testdrehbuch	Bei einer nicht existierende Flugbuch DB gibt es eine Exception.	Testversion 0.1
Testversion 0.0	Testdrehbuch	Flugzeugtyp ohne Namen können generiert werden. → In Flugbuch DB wird ein leerer Eintrag generiert jedoch wird keine Flugzeugtyp-Datei erzeugt.	Testversion 0.1
Testversion 0.1	Testdrehbuch	Gleichnamige Flugzeugtypen können erfasst werden. → in der DB erscheinen dann zwei Einträge obwohl nur eine Datei existiert.	Testversion 0.2
Testversion 0.1	Testdrehbuch	Bemerkungen im Flugbuch können zu lange beschrieben werden (Überlauf auf der Konsole)	Testversion 0.2
Testversion 0.1	Testdrehbuch	Wenn die Periode im Flugbuch mit F2 korrigiert wird, werden die geänderten Daten nicht in die Formular-Datei geschrieben.	Testversion 0.2
Testversion 0.1	Testdrehbuch	Gleiches Problem wie letzter Test allerdings bei der Flugplanung.	Testversion 0.2
Testversion 0.2	Benutzer Test	Schreibfehler an diversen Orten	Testversion 0.3
Testversion 0.2	Benutzer Test	Ungeschickte Farbwahl	Testversion 0.3
Testversion 0.2	Benutzer Test	Tab-Taste sollte bei der Eingabe benötigt werden können.	Testversion 0.3
Testversion 0.2	Benutzer Test	Korrektur der vorhergehenden Felder sollte möglich sein. Auch wenn diese dabei gelöscht würden.	Testversion 0.3
Testversion 0.2	Benutzer Test	Bemerkungen teilweise verwirrend bei den Eingabe Feldern	Testversion 0.3
Testversion 0.2	Benutzer Test	TAS sollte als Vorgabe vorgegeben werden (Flugplanung).	Testversion 0.3
Testversion 0.3	Testdrehbuch	Probleme bei der Installation wegen der fehlenden *.ocx Datei.	Fehler bei der Erstellung des Setup.exe
Testversion 0.4	Benutzer Test	Applikation läuft auf Teile von WIN NT nicht.	Wird nicht behoben

9 Anhang

9.1 Definitionen, Abkürzungen

ALTN	Alternate (Ausweich Flugplatz)
BAZL	Bundesamt für Zivilluftfahrt
COPI	Ko- Piloten
PIC	Pilot in comand
SIM	Simulator

9.2 Referenz und Literatur

[1]	www.devmentor.ch	Homepage M.Burri
[2]	www.c-plusplus.de	Entwickler Homepage C++
[3]	C++ lernen und professionell anwenden	Peter Prinz, Ulla Kirch-Prinz
[4]	Easy C++	Markt + Technik Verlag

9.3 Projektabwicklung

Datum	**Tätigkeit**
06.11.2002	Besprechungsgespräch 1 Grundlagen und Pflichtenheft mit D. Forrer
06.11.2002	Besprechungsgespräch 1 Konzeptionelles mit D. Forrer
09.01.2003	Besprechungsgespräch 2 mit D. Forrer
11.02.2003	Dokumentation Rewiew mit D.Forrer

9.4 Werdegang dieses Dokuments

Version	**Datum/ Tätigkeiten**
V0.01	10. November 02/ Grundstruktur erstellt
V0.20	11. November 02/ Pflichtenheft erstellt
V0.30	02. Dezember 02/ Soll-/ Ist Vergleich eingefügt
V0.40	26. Dezember 02/ Konzept schreiben
V0.50	03 bis 05 Januar 03/ Dokumentation Komplett nachführen inklusive Implementation (erster Entwurf)
V0.60	20 bis 30 Januar 03/ Dokumentation Korrigieren lassen
V1.00	17.02.03/ Dokumentation fertigstellen/ Abgabefertig

9.5 Testdrehbuch

Das Testdrehbuch soll dazu beitragen, eine hohe Softwarequalität vor Abgabe der ersten Software Version zu erreichen. Wie schon erwähnt wurde dieses Testdrehbuch mehrmals mit mehreren Testversionen durchgespielt. Ebenso wurden Erfahrungen aus den Benutzer-Test zugefügt. Die Resultate der Tests in diesem Formular beziehen sich auf den letzten Testdurchlauf vor der Abgabe.

9.5.1 Voraussetzungen

Folgende Hardwarekonfiguration garantieren die positiven Resultate der Tests.

- Windows 2000 Proffesional Service Pack 3
- Windows XP Home Edition Service Pack 1
- Pentium > 300 MHz
- RAM 128 MB oder grösser
- Bildschirmauflösung 1024 x 768

Es wird hier nochmals darauf hingewiesen, dass die Applikation auf Windows 98 nicht funktionsfähig ist und dass sie nicht auf Windows NT getestet wurde!

9.5.2 Installation/ Grundfunktionen

Test-fall	Eingabe / Aktion	Reaktion	Resultat
1	Applikation mit Setup-programm, dass sich auf der CD befindet installieren.	Software wird ohne Fehler installiert. Die Ordner wurden ins Installationspfad installiert. Das Icon erscheint auf dem Desktop und im Start Menü von Windows.	✓
2	Überprüfen ob OCX Datei im System32 Ordner installiert wurde.	Comdlg32.ocx Datei wurde installiert.	✓
3	Programm über Desktop starten.	Programm startet ohne Fehlermeldung.	✓
4	Programm über Start Menü starten.	Programm startet ohne Fehlermeldung.	✓
5	Navigieren in den Menüs.	Es kann in den Menüs navigiert und jederzeit mit ESC einen Schritt zurück gegangen werden.	✓
6	Flugbuch.dtb Datei löschen und Programm starten. Flugbuch wählen und neuen Eintrag eingeben.	Eine neue Datenbank wird erstellt. Der erste und letzte Eintrag wird darin erzeugt. Nach der Eingabe eines neuen Eintrages befindet sich dieser auch in der erzeugten Datenbank.	✓
7	Beenden der Applikation.	Die Applikation kann im Hauptmenü mit der ESC Taste beendet werden.	✓

9.5.3 Flugbuch Daten erfassen

Test-fall	Eingabe / Aktion	Reaktion	Resultat
1	Flugbucheingabe wählen.	Eingabemenü wird angezeigt. Cursor steht auf Datumseingabe.	✓
2	Falsches Datumsformat eingeben und ENTER drücken.	Wird nicht akzeptiert. Erneute Eingabemöglichkeit wird gewährt.	✓
3	Dateneingeben bis zur Flugarteingabe.	Alle Daten können eingegeben werden. Keine Überprüfung der Felder.	✓
4	Flugart eingeben und eine falsche Eingabe machen und ENTER drücken.	Wird nicht akzeptiert. Erneute Eingabemöglichkeit wird gewährt.	✓
5	Daten eingeben bis zur Startzeit.	Alle Daten können eingegeben werden. Keine Überprüfung der Felder.	✓
6	Startzeit eingeben und Format falsch eingeben und ENTER drücken.	Wird nicht akzeptiert. Erneute Eingabemöglichkeit wird gewährt.	✓
7	Startzeit nach Vorgabe eingeben und ENTER drücken.	Startzeit wird akzeptiert.	✓
8	Testfälle 6 und 7 wiederholen mit Landezeit.	Gleiches Verhalten.	✓
9	Anzahl Landungen eingeben jedoch Buchstaben anstatt Zahlen und ENTER drücken.	Wird nicht akzeptiert. Erneute Eingabemöglichkeit wird gewährt.	✓
10	Anzahl Landungen mit Zahlen eingeben.	i.O	✓
11	Formular fertig ausfüllen.	Abfrage erscheint ob gespeichert werden soll oder nicht.	✓
12	F2 drücken.	Daten werden verworfen (keine Speicherung erfolgt).	✓
13	Testfälle 1 bis 11 wiederholen ohne Falsch-eingaben. Und F1 drücken.	Datensatz wurde abgespeichert.	✓
14	Testfälle 1 bis 11 wiederholen mit mehreren Datums. (jeden Eintrag speichern)	Einträge werden nach Datum in die Datenbank eingefügt. Bei mehreren gleichen Datums wird der letzte Eintrag am Schluss der Kolonne abgespeichert.	✓
15	Testfälle wiederholen und Backspace-Funktion überprüfen.	1 mal drücken → aktuelles Feld wird gelöscht. 2 mal drücken → letztes Feld wird gelöscht.	✓

9.5.4 Flugbuchdaten löschen

Test-fall	Eingabe / Aktion	Reaktion	Resultat
1	Im Flugbuch Lösch-Menü anwählen.	Lösch Menü wird angezeigt. Datenbankinhalt wird angezeigt.	✓
2	Mehr als 10 Datensätze generieren. Und wieder ins Lösch-Menü einsteigen.	Mit den Pfeiltasten „Hoch/ Runter“ kann in der Datenbank navigiert werden.	✓
3	Sich einen Eintrag merken bzw. dessen Index und F1 drücken. Index eingeben der gelöscht werden soll und ENTER drücken.	Datensatz wurde aus der Datenbank entfernt.	✓
4	Test 3 wiederholen und Buchstaben anstatt Zahlen eingeben und mit ENTER bestätigen.	Es darf nichts gelöscht werden.	✓
5	Test 3 wiederholen und 0 oder einen grösseren Index als vorhanden eingeben.	Es darf nichts gelöscht werden.	✓

9.5.5 Flugbuchdaten Filtern

Test-fall	Eingabe / Aktion	Reaktion	Resultat
1	Flugzeitnachweis erstellen wählen.	Flugzeitnachweis Menü wird angezeigt. Die gesamte Flugerfahrung wird angezeigt. Berechnete Daten von Hand überprüfen (sie müssen identisch sein).	✓
2	„Von“-Datum eingeben aber in einem falschen Format.	Wird nicht akzeptiert. Erneute Eingabemöglichkeit wird gewährt.	✓
3	Von“-Datum diesmal richtig eingeben.	Eintrag wird akzeptiert.	✓
4	Test 2 und 3 wiederholen jedoch mit dem „Bis“ Datum.	Identisches Verhalten.	✓
5	Nach Eingabe des „Bis“ ENTER drücken.	Gefilterte Periode wird auf der rechten Seite angezeigt. Manuell nachrechnen und überprüfen. Taskmenü erscheint ob gedruckt werden soll oder nicht.	✓
6	F2 drücken.	Daten werden verworfen.	✓
7	Tests 1 bis 6 wiederholen ohne Falscheingaben.	In jedem Fall sind die berechneten Daten korrekt.	✓

9.5.6 Flugzeitnachweis-Formular

Test-fall	Eingabe / Aktion	Reaktion	Resultat
1	Nach der Eingabe eines „Von“ und „Bis“ Datum. F1 drücken.	Formular wird erstellt (Info wird angezeigt). Visual Basic Applikation wird gestartet.	✓
2	Nach einem Augenblick.	Fenster „Vorschau/ Drucken“ erscheint.	✓
3	Vorschau drucken.	Das Formular wird angezeigt. Daten stimmen mit den berechneten Werten überein.	✓
4	Testsequenz 1 bis 3 wiederholen mit diversen Eingaben.	Formular stimmt immer mit berechneten Daten überein.	✓
5	Drucken wählen.	Auswahl Menü des Druckers erscheint.	✓
6	Drucker auswählen und OK drücken.	Formular wird 1 zu 1 ausgedruckt. Alle Parameter sind gut lesbar.	✓
7	Beide Druckfunktionen (Auswahlfenster und Formularfenster) einmal betätigen.	Jedes Mal erscheint das Drucker-auswahl Fenster. Die Formulare können auf beide Arten ausgedruckt werden.	✓
8	Alle Fenster der Druck-applikation beenden.	Fenster werden geschlossen.	✓

9.5.7 Flugzeugtyp erfassen

Test-fall	Eingabe / Aktion	Reaktion	Resultat
1	Neuen Flugzeugtyp erfassen wählen. Call Sign eingeben.	Menü wird korrekt angezeigt und Cursor steht auf dem Call Sign.	✓
2	Alle Parameter eingeben und ENTER drücken.	Auswahl erscheint ob der Flugzeugtyp gespeichert werden soll oder nicht.	✓
3	Eintrag mit F1 Speichern	Datensätze werden gespeichert. Datei wird erzeugt und Eintrag in die FlugzeugtypPfad.dtb wird generiert.	✓
4	Test 1 Wiederholen mit dem gleichen Namen.	Benutzer wird darauf hingewiesen, dass der Flugzeugtyp schon existiert. Die Eingabe wird verworfen.	✓
6	Test 1 wiederholen ohne etwas einzugeben und ENTER drücken.	Nichts passiert. Cursor bleibt stehen.	✓
7	Tests 1 und 2 wiederholen und irgendwo die Backspace-Taste drücken.	1 mal drücken → Aktuelles Feld wird gelöscht. 2 mal drücken → letztes Feld wird gelöscht. Eine neue Eingabe des entsprechenden Feldes ist dann möglich.	✓

Test-fall	Eingabe / Aktion	Reaktion	Resultat
5	Test 1 wiederholen bis 10 verschiedene Flugzeugtypen in der Datenbank eingetragen sind.	Es existieren 10 Flugzeugtyp Dateien und 10 Einträge in der FlugzeugtypPfad.dtb.	✓
6	Test 1 Wiederholen und einen 11 Flugzeugtyp generieren.	Benutzer wird darauf hingewiesen, dass schon 10 Typen existieren (können nicht noch mehr generiert werden), Eingabe wird verworfen.	✓

9.5.8 Flugzeugtyp bearbeiten

Test-fall	Eingabe / Aktion	Reaktion	Resultat
1	Flugzeugtyp Bearbeiten wählen.	Menü wird korrekt angezeigt.	✓
2	Einen Flugzeugtyp wählen und ENTER drücken.	Alle Parameter die erfasst sind werden angezeigt.	✓
3	Jeden Parameter wählen, verändern und speichern.	Jeder Parameter lässt sich verändern ausser der Call Sign (File Name).	✓
4	Test 3 wiederholen jedoch jedesmal nein drücken.	Parameter werden nicht verändert.	✓
5	Test 1 bis 5 wiederholen mit allen 10 Flugzeugtypen.	Gleiche Resultate wie vorhergehende Tests.	✓

9.5.9 Flugzeugtyp löschen

Test-fall	Eingabe / Aktion	Reaktion	Resultat
1	Flugzeugtyp Löschen wählen.	Alle erfasste Flugzeugtypen erscheinen.	✓
2	Einen Flugzeugtyp wählen und ENTER drücken.	Flugzeugtyp wird nach einer Sicherheits-abfrage gelöscht.	✓
3	Test 1 und 2 wiederholen bis alle gelöscht sind.	Keine Dateien mehr vorhanden, die FlugzeugtypPfad.dtb Datenbank ist leer.	✓
4	Flugzeugtyp Bearbeiten wählen.	Der Benutzer wird darauf hingewiesen, dass kein Flugzeugtyp in der Datenbank ist.	✓
5	Flugzeugtyp Löschen wählen.	Der Benutzer wird darauf hingewiesen, dass kein Flugzeugtyp in der Datenbank ist.	✓

9.5.10 Flugplanung erstellen

Test-fall	Eingabe / Aktion	Reaktion	Resultat
1	Flugplanung erstellen wählen.	Der Benutzer wird darauf hingewiesen, dass kein Flugzeugtyp in der Datenbank ist.	✓
2	Flugzeugtyp erfassen.	Daten werden übernommen und Dateien erstellt.	✓
3	Flugplanung erstellen wählen.	Flugzeugtyp kann jetzt ausgewählt werden (1 in der Auswahl)	✓
4	Flugzeugtyp auswählen und ENTER drücken.	Gewichterfassungs Menü wird angezeigt.	✓
5	Gewichte angeben. Backspace-Funktion überprüfen.	Gewichte können erfasst werden. Die Backspace-Taste funktioniert identisch wie im Flugbuch.	✓
6	Nach der Gewichts-erfassung ENTER drücken.	Flugplanungs Formular wird als Menü angezeigt.	✓
7	Start Flugplatz erfassen. Backspace-Funktion überprüfen.	Daten können eingegeben werden. Backspace-Funktion wie beim Flugbuch.	✓
8	Ersten Routenpunkt erfassen. Bis zum Ground Speed.	Alle Parameter können eingegeben werden. Backspace-Taste funktioniert wie beim Flugbuch. Nach der Eingabe der Distanz wird die TAS des ausge-wählten Flugzeugtyps in das GS- Feld eingetragen (rot)	✓
9	GS überschreiben.	Der GS kann überschrieben werden.	✓
10	Bemerkung eingeben und ENTER drücken.	Das Task-Menü wird aktiv. F1 um noch einen Eintrag einzugeben, F2 um die Zeile zu löschen und F3 um den Alternate-Flugplatz einzugeben.	✓
11	Tests 1 bis 10 wiederholen und F1 bis F3 ausprobieren.	Es passiert immer das was angeschrieben ist.	✓
12	Test 1 bis 10 wiederholen und alle 15 Felder eingeben.	Alle Parameter können eingegeben werden. Wenn beim GS ENTER gedrückt wird, wird der TAS Wert übernommen.	✓
13	Alternate Flugplatz erfassen. (Analog zur Route) und Enter drücken.	Die Berechnungen (Weight und Balance/ Fuel Calculation) werden durchgeführt und auf der Konsole angezeigt.	✓
14	Flugplanung von Hand durchführen und Resultate überprüfen.	Resultate sind identisch.	✓
15	Nach dem Alternate hat der Benutzer die Auswahl, das Formular zu erstellen oder die Daten zu verwerfen. F2 drücken.	Daten werden verworfen. Eingabe kann nochmals durchgeführt werden.	✓

Test-fall	Eingabe / Aktion	Reaktion	Resultat
16	Test 1 bis 14 wiederholen mit mehr oder weniger Einträgen.	Berechnungen sind immer korrekt.	✓
17	Backspace überprüfen.	Während der Eingabe einer Zeile kann jeweils das Eigene und das letzte Feld gelöscht werden (Um eine ganze Zeile zu löschen muss die F2 Taste gedrückt werden).	✓

9.5.11 Navigation Flugplan Formular

Test-fall	Eingabe / Aktion	Reaktion	Resultat
1	Nach der Eingabe einer kompletten Flugplanung F1 drücken.	Formular wird erstellt (Info wird angezeigt). Visual Basic Applikation wird gestartet.	✓
2	Nach einem Augenblick.	Fenster „Vorschau/ Drucken“ für Navigation-Flugplan und Zusatzformular erscheint.	✓
3	Auf die Vorschau des Navigation Flugplan drücken.	Das Formular wird angezeigt. Die Daten stimmen mit den eingesetzten überein die. Das Formular entspricht einem Standard-Flugplanungsformular.	✓
4	Testsequenz 1 bis 3 wiederholen mit diversen Flugplanungen (wenig und viele Zwischenpunkte).	Formular entspricht immer mit den eingegebenen Daten überein.	✓
5	Drucken wählen.	Auswahlmenü des Druckers erscheint.	✓
6	Drucker auswählen und OK drücken.	Formular wird 1 zu 1 ausgedruckt. Alle Parameter sind gut lesbar.	✓
7	Beide Druckfunktionen (Auswahlfenster und Formularfenster) einmal betätigen.	Jedes Mal erscheint das Drucker-auswahl-Fenster. Die Formulare können auf beide Arten ausgedruckt werden.	✓

9.5.12 Zusatzformular

Test-fall	Eingabe / Aktion	Reaktion	Resultat
1	Auf die Vorschau des Zusatzformulars drücken	Das Formular wird angezeigt. Die berechneten Daten stimmen überein.	✓
2	Test 1 mit diversen Flugplanungen durchführen.	Formular und berechnete Daten stimmen immer überein mit den berechneten Daten der Applikation.	✓
5	Drucken wählen.	Auswahlmenü des Druckers erscheint.	✓
6	Drucker auswählen und OK drücken.	Formular wird 1 zu 1 ausgedruckt. Alle Parameter sind gut lesbar.	✓
7	Beide Druckfunktionen (Auswahlfenster und Formularfenster) einmal betätigen.	Jedes Mal erscheint das Drucker-auswahl-Fenster. Die Formulare können auf beide Arten ausgedruckt werden.	✓
8	Alle Fenster der Druck-applikation beenden.	Fenster werden geschlossen.	✓

9.5.13 Hilfe

Test-fall	Eingabe / Aktion	Reaktion	Resultat
1	F10 im Hauptmenü drücken.	Hilfe zum Hauptmenü erscheint.	✓
2	ESC drücken.	Die Applikation ist wieder im Hauptmenü.	✓
3	F10 im Flugbuch Teil drücken.	Hilfe zum Flugbuch erscheint.	✓
4	ESC drücken.	Die Applikation ist wieder im Flugbuch-menü	✓
5	F10 im Flugplanungsteil drücken.	Hilfe zur Flugplanung erscheint.	✓
6	ESC drücken.	Die Applikation ist wieder im Flugplanungmenü	✓
7	Navigieren in allen Menüs ist mit den Pfeiltasten „Hoch/ Runter“ möglich.	i.O	✓

9.5.14 Allgemein

Allgemeine Tests sind Benutzertests, die ohne System durchgeführt wurden. In diesem Kapitel sind darum nur die problematischen Testfälle erfasst worden.

Test-fall	Eingabe / Aktion	Reaktion	Resultat
1	Schnelles Betätigen der Taste ENTER.	Keine Exception darf auftreten.	✓*
2	Drücken aller möglichen Tastenkombinationen und Zeichen während die Applikation läuft.	Keine Exception darf auftreten.	✓*

* Die Applikation ist bei diesen Tests nie abgestürzt, jedoch z.T. hängen geblieben. das Drücken der ESC Taste (auch wenn mehrmals) hat das Problem jeweils gelöst.

9.5.15 Resultate

Die Testreihe hat gezeigt, dass die Version 1.0 funktionsfähig ist. Um Stabilität im täglichen Gebrauch zu erlangen, wurden die Testreihen umfangreich durchgeführt, als dies ein Benutzer tun würde. Die Benutzertests sind mehrheitlich positiv ausgefallen. Die rückgemeldeten Verbesserungsvorschläge wurden bereits vorgängig in die Applikation implementiert. Die Applikation kann somit freigegeben werden.

9.6 Dateienbeschreibung

In diesem Abschnitt werden die Dateien und deren Aufbau beschrieben. Der Leser weiss nachher welcher Parameter abgespeichert ist um eine Zusatzentwicklung realisieren zu können.

9.6.1 Flugbuch.dtb

Die Flugbuchdatenbank beinhaltet Einträge zum Flugbuch. Jeder Flugbucheintrag benötigt eine Zeile. Die Datenbank ist durch einen Start und einen End-Wert eingegrenzt.

```
0.00.0000,erster Eintrag,,,,,,,,,,,
Datum, Typ, Imma., Flugart, Start Flp., Lande Flp., Start Zeit, Lande Zeit, PIC, COPI , SIM , Lndg., FL, Bemerkung
.....
.....
99.99.9999,letzter Eintrag,,,,,,,,,,,
```

Abbildung 19: Darstellung der Flugbuch.dtb Datei

Blau: Erster und letzter Eintrag. In diesem Bereich kann die Applikation filtern
Rot: Datensätze (Komma getrennt)

9.6.2 Flugzeitnachweis.dta

Die Flugzeitnachweisdatei (Flugzeitnachweis.dta) wird erzeugt um die Formular ausgabe zu gewährleisten. Aus dieser Datei liest die Visual Basic Applikation (FormularFlugbuch.exe) die Datensätze aus, um das Formular zu erstellen.

```
Von Datum
Nach Datum
Gesamte Flugerfahrung
Flugerfahrung PIC
Flugerfahrung COPI
Flugerfahrung SIM
Gesamte Anzahl Landungen
Gesamte Flugerfahrung in der gefilterten Periode
Flugerfahrung PIC in der gefilterten Periode
Flugerfahrung COPI in der gefilterten Periode
Flugerfahrung SIM in der gefilterten Periode
Gesamte Anzahl Landungen in der gefilterten Periode
```

Abbildung 20: Darstellung der Flugzeitnachweis.dta Datei

9.6.3 FlugzeugtypenPfad.dtb

Die Flugzeugtyppfad Datenbank (FlugzeugtypPfad.dtb) beinhaltet lediglich die Filenamen der Flugzeugtypen (File Name = Immatrikulation). Diese Datei wird benötigt um dem Benutzer die verfügbaren Dateien anzuzeigen.

```
FileName1
FileName2
FileName3
FileName4
FileName5
.....
FileName10
```

Abbildung 21: Darstellung der FlugzeugtypenPfad.dtb Datei

Die Datensätze werden beim Einfügen oder löschen nicht sortiert.

9.6.4 Flugzeugtypen (*.dta)

Die Flugzeugtypen-Dateien (*.dta) beinhalten die spezifischen Flugzeugtypen. Es gibt maximal 10 Dateien dessen Filenamen in der FlugzeutypenPfad.dtb Date abgespeichert sind. Aus dieser Datei liest die Visual Basic Applikation (FormularFlugplanung.exe) einen Teil der Datensätze aus, um das Zusatzformular (Weight & Balance/ Fuel Calculation) zu erstellen.

```
Type
Call Sign (= File Name)
TAS
Fuel
Fuel Flow
Bew
MTOW
Bew Arm
Crew Arm
Pax Arm
Bag Arm
Fuel Arm
```

Abbildung 22: Darstellung der Flugzeugtyen Dateien

9.6.5 StartFlugplatz.dta

Die Startflugplatz-Datei (StartFlugplatz.dta) beinhaltet den für die Flugplanung eingegebenen Start-Flugplatz. Aus dieser Datei liest die Visual Basic Applikation (FormularFlugplanung.exe) einen Teil der Datensätze aus, um den Navigationsflugplan zu erstellen.

```
Frequenz Startflugplatz
Name Startflugplatz
Bemerkungen Startflugplatz
```

Abbildung 23: Darstellung der StartFlugplatz.dta Datei

9.6.6 Route.dta

Die Routen-Datei (Route.dta) beinhaltet die Route der Flugplanung (max. 15 Einträge). Aus dieser Datei liest die Visual Basic Applikation (FormularFlugplanung.exe)) einen Teil der Datensätze aus, um den Navigationsflugplan zu erstellen.

```
Frequenz, Check Punkt, MT, Distanz, GS, EET, Bemerkungen
.....
.....
.....
```

Abbildung 24: Darstellung der Route.dta Datei

9.6.7 Alternate.dta

Die Alternate-Datei (Alternate.dta) beinhaltet den für die Flugplanung definierten Alternate Flugplatz. Aus dieser Datei liest die Visual Basic Applikation (FormularFlugplanung.exe) einen Teil der Datensätze aus, um den Navigations Flugplan zu erstellen.

```
Alternate Frequenz
Alternate Check Punkt
Alternate Magnetical Track (MT)
Alternate Distanz
Alternate Ground Speed
Alternate Flugzeit
Alternate Bemerkungen
```

Abbildung 25: Darstellung der Alternate.dta Datei

9.6.8 Berechnungen.dta

Die Berechnung-Datei (Berechnungen.dta) beinhaltet die für die Flugplanung berechneten Daten. Aus dieser Datei liest die Visual Basic Applikation (FormularFlugplanung.exe) einen Teil der Datensätze aus, um das Zusatzformular (Weight & Balance/ Fuel Calculation) zu erstellen.

Berechnete Daten FuelTrip
Berechnete Daten FuelAdditional
Berechnete Daten FuelTotal
Berechnete Daten FlugZeitTrip
Berechnete Daten FlugZeitAdditional
Berechnete Daten FlugZeitTotal
Berechnete Daten DistanzTrip
Berechnete Daten DistanzAlternate
Berechnete Daten FlugzeitTypIndex (wird benötigt um richtigen Flugzeugtyp auszulesen)

Abbildung 26: Darstellung der Berechnungen.dta Datei

9.7 CD

Auf der CD sind folgende Dateien vorhanden:

- Setup Programm
- Code C++
- Code Visual Basic
- Dokumentation
- Bedienungsanleitung

Auf der folgenden Abbildung sind die Dateitypen die auf der CD zu finden sind, kurz beschrieben.

Erweiterung	Text	Binär	Beschreibung
.cpp		x	Source Code Programmiersprache C++
.h		x	Header Datei Programmiersprache C++
.dsp		x	Datei von Visual C++ erzeugt
.dsw		x	Datei von Visual C++ erzeugt
.opt		x	Datei von Visual C++ erzeugt
.plg		x	Datei von Visual C++ erzeugt
.aps		x	Datei von Visual C++ erzeugt
.rc		x	Datei von Visual C++ erzeugt
.ico		x	*Icon*, Datei enthält ein Windows-Icon
.obj		x	*object file*, kompilierter Programmcode
.pch		x	Datei von Visual C++ erzeugt
.res		x	Datei von Visual C++ erzeugt
.idb		x	Datei von Visual C++ erzeugt
.ilk		x	Datei von Visual C++ erzeugt
.bas		x	Datei von Visual Visual Basic erzeugt
.vpb		x	Datei von Visual Visual Basic erzeugt
.vpw		x	Datei von Visual Visual Basic erzeugt
.frm		x	Datei von Visual Visual Basic erzeugt
.log	x		*Log*, Protolldatei
.scc		x	Datei von Visual Visual Basic erzeugt
.ocx		x	Datei von Visual Visual Basic erzeugt
.exe		x	*executable*, ausführbares Programm
.dtb	x		Textdatei (Datenbank Flugplaner)
.dta	x		Textdatei (Datei Flugplaner)
.txt	x		Textdatei (Hilfe Datei Flugplaner)
.pdf	x		Acrobat-Datei (geschützt)
.doc	x	x	*document*, Word-Dokument oder ASCII-Text
.zip	x		Archivdatei von *Pkzip.exe* oder kompatibel

Abbildung 27: Liste der Erweiterungen (Dateitypen)

Hinweis:

→ Um eine Überprüfung der Applikation zu ermöglichen habe ich die Datenbanken schon mit Einträge abgefüllt. Diese werden bei der Installation der Applikation automatisch installiert.