Duale Hochschule Baden-Württemberg

Mannheim

Erste Projektarbeit

Erarbeitung eines Lösungsentwurfes für eine IT-Lösung zur Kapazitätsplanung der Kabinencrew

Studiengang Wirtschaftsinformatik - Sales & Consulting

Bearbeitungszeitraum: 15.05.2017 - 29.08.2017

Verfasser:	Julian Garske
Matrikelnummer:	6728241
Kurs:	WWI SCA16
Studiengangsleiter:	Dr. Frank Koslowski
Wissenschaftlicher Betreuer:	Günter Stumpf
Telefon:	01511 8237778
Mailadresse:	guenter.stumpf@esosec.de
Ausbildungsbetrieb:	Lufthansa Systems GmbH & Co. KG Am Prime Parc 1 D 65479 Raunheim
Unternehmensbetreuer:	Berger, Iwan
Telefon(Firma):	+49 (0)69 696 74135
Mailadresse(Firma):	iwan.berger@lhsystems.com

Inhaltsverzeichnis

Kurzfassung (Abstract)

Abkürzungsverzeichnis

BT Beschäftigungstage

CAB Compas Cabin

CDB Crew Database

CMS Crew Management System

COC Compas Cockpit

KG Kleingruppe

PU Planungseinheit

Abbildungsverzeichnis

Anlagenverzeichnis

1 Einleitung

1.1 Motivation

Bereits seitdem es kommerzielle Passagierflüge gibt, ist eine konkrete Zuteilung der Besatzung für die Flüge notwendig. Dabei müssen nicht nur Fehlzeiten wie Urlaub oder Krankheit, sondern auch andere Faktoren wie z.B. Teilzeit berücksichtigt werden. Außerdem werden für unterschiedliche Flugzeuge auch unterschiedliche Qualifikationen benötigt. Bei immer größer werdenden Fluggesellschaften, wie der Lufthansa mit insgesamt ca. 20.000 Besatzungsmitgliedern der Kabine, stellt die Zuteilung aus diesen Gründen oft eine Herausforderung dar. Die sogenannte Kapazitätsplanung ist deshalb für viele Airlines ein wichtiger Bestandteil des Flugbetriebs.
Mit der Kapazitätsplanung hängt die Schulungsplanung eng zusammen. Schulungen dauern oft einige Wochen und müssen daher langfristig vorher geplant werden, sodass Fehlzeiten ausgeglichen werden können und die Qualifikationen nach der Schulung aktualisiert werden. Das Ziel der Kapazitäts- und Schulungsplanung ist, „durch rechtzeitige Neueinstellungen und Umschulungen die richtige Menge an Piloten mit der richtigen Qualifikation zum richtigen Zeitpunkt auf einer Flotte bereitzustellen“ (vgl. Berger, 2016, S.19).

Für die Kapazitäts- und Schulungsplanung der Cockpit-Besatzung im Lufthansa Konzern gibt es seit 2000 das von Lufthansa Systems entwickelte System Compas Cockpit (COC). Dadurch ist eine Planung der Einteilung und Schulungen für die etwa 5.000 Piloten bis zu 15 Monate in Zukunft möglich. Bis heute wird die Funktionalität dieses Programms regelmäßig erweitert.
Mit der Entwicklung von COC entstand eine Vorstellung, ein ähnliches Programm auch für die Kabinenbesatzung zu entwickeln. Durch diese systematische Lösung sollen Zeit gespart und Fehler minimiert werden (vgl. Winkel, 2017, S.4). Ein Auftrag für das Projekt wurde von dem Kunden, der Lufthansa Passage Airline, noch nicht veröffentlicht, weshalb noch nicht zu einem Projekt mit einer konkreten Analyse oder Entwicklung kam.

Letztes Jahr hatte sich die Firma M2P Consulting bereits mit der bestehenden Kapazitätsplanung auseinandergesetzt und geprüft, wie man diese verbessern könne und ob COC dafür in Frage käme. Sie kamen zu dem Ergebnis, dass kurzfristig zwar die bisherige Vor-

gehensweise der Kapazitätsplanung (siehe 1.6) optimiert werden könne, um die Qualität zu verbessern, langfristig solle es aber durch eine systemseitige Lösung abgelöst werden (vgl. M2P, 2016, S.8-10).

1.2 Problemstellung und -abgrenzung

Zurzeit werden die Einsätze der Kabinenbesatzung mithilfe von verschiedenen Excel-Tabellen geplant. Diese ca. 200 Tabellen enthalten Daten aus unterschiedlichen Quellen, die für die Zuteilung der Besatzung erforderlich sind. Aufgrund der großen Datenmenge und den Verflechtungen der Tabellen untereinander kommt es oft zu Problemen und Fehlern bei der Kapazitätsplanung (vgl. Berg u. a., 2017).

Nach dem Vorbild von COC soll eine automatisierte Kapazitäts- und Schulungsplanung jetzt auch für die Kabine, also die komplette Crew, unter dem Titel Compas Cabin (CAB) ermöglicht werden. Dabei gilt es, zu ermitteln wie weit COC dafür als Vorlage genutzt werden kann. Im Vordergrund steht dabei erst einmal die Bestandsrechnung, die in dieser Arbeit angepasst wird. Sie ist Grundlage für die Kapazitätsplanung, in der sie mit dem Bedarf ins Verhältnis gesetzt wird.
Als Ausgangspunkt für das Projekt wird COC genutzt, da es viele gewünschten Funktionen schon für die Cockpit-Besatzung enthält. Einige Unterschiede, die es zwischen der Kapazitätsplanung der Cockpit- und der der Kabinenbesatzung gibt, müssen analysiert und geklärt werden, um zu entscheiden, ob das gewünschte Programm in COC integriert werden kann. Diese Unterschiede werden im Kapitel 2 genauer erläutert.

Besonderer Fokus liegt auf der Integration in die bestehende Systemlandschaft von COC und die Entwicklung einer mandantenfähigen Lösung, um mehrere Airlines der Lufthansa Gruppe zu integrieren. Trotz der Integration soll das Programm mit Blick auf die Zukunft moderne Architekturansätze beinhalten und sich von den fast 20 Jahre alten von COC loslösen.

1.3 Ziel der Arbeit

Das Ziel dieser Projektarbeit ist, einen initialen Lösungsentwurf für die Entwicklung eines explorativen Prototyps zur Kapazitätsplanung des Kabinenpersonals zu erstellen. Daraus sollen vor allem die Anforderungen und Funktionen erkennbar sein. Der explorative Prototyp soll nach seiner Entwicklung evolutionär genutzt werden und damit Ausgangspunkt für Erweiterungen sein.
Auf der Grundlage des Prototyps soll die sogenannte Deltarechnung ermöglicht werden. Dabei wird der aktuelle Personenbestand mit dem Bedarf ins Verhältnis gesetzt und damit die Kapazitäten ermittelt. Für zukünftige Prognosen werden Prämissen und Erfahrungswerte genutzt.

In dieser Arbeit wird für die Kapazitätsplanung die Bestandsrechnung erläutert und angepasst. Darüber hinaus wird evaluiert, ob und wie weit COC dazu als Vorlage dienen kann und welche Unterschiede in der Bestandsrechnung zwischen der Cockpit- und der Kabinenbesatzung beachtet werden müssen. Für Probleme, die aus diesen Unterschieden resultieren, soll eine Lösung gefunden werden, damit das Modell von COC in ähnlicher Weise auf die Kapazitätsplanung der Kabine angewendet werden kann.

1.4 Vorgehen

Zunächst ist es nötig, die Anforderungen zu ermitteln und abzuklären. Dafür muss die aktuelle Vorgehensweise der Kapazitätsplanung verstanden und die Anforderungen, die das zu entwickelnde Programm erfüllen soll, erfasst werden. Die Anforderungen werden in diesem Projekt hauptsächlich von den momentanen Planerinnen, die das spätere Programm anwenden, gestellt.
Danach werden Gemeinsamkeiten und Unterschiede zwischen CAB und COC ermittelt, um damit bewerten zu können, wie COC als Vorlage geeignet ist. Eine Anwenderdokumentation dient als Grundlage für die Entwicklung des Prototyps. Sie beschreibt aus fachlicher Sicht, wie der Nutzer mit dem Programm interagieren kann und welche Funktionen es beinhaltet.

1.5 Überblick über Compas Cockpit

COC (Crew Operation Manpower Planning Advanced System) dient als Ausgangspunkt für das Projekt und nur einige Besonderheiten müssen bei dem aktuellen Programms angepasst werden, um die Kapazitätsplanung der Kabinenbesatzung dadurch zu automatisieren (vgl. Wagner u. a., 2017b). Deshalb sind ein Überblick über COC und Kenntnisse der wichtigsten Funktionen davon wichtig. Dieser Überblick stammt aus einem betriebsinternen Dokument (vgl. Berger, 2016).
COC ist das zentrale Tool für die Kapazitätsplanung des Cockpitpersonals. Ziel dieser Planung ist es, die richtige Anzahl an Piloten mit passenden Qualifikationen zu einem bestimmten Zeitpunkt auf eine Flotte bereitstellen zu können. Das kann kurzfristig gesteuert oder langfristig prognostiziert werden. Diese Planung ist sehr komplex, da sehr viele Einflussfaktoren dort berücksichtigt werden müssen.
Um während der Planung den Überblick zu behalten, teilt COC jeden Piloten eindeutig in eine Planungseinheit (PU) ein, die durch Flugzeugtyp, Flotte, Standort, Funktion und Unterfunktion definiert ist. Dadurch kann für jede PU für einen bestimmten Zeitraum tagesgenau z.B. der Bestand und Bedarf ermittelt und verglichen werden.

Zusätzlich wurde das Programm mehrmals erweitert, sodass es mittlerweile aus vier internen Hauptmodulen, z.B. einem Schulungsplaner und der Deltarechnung und zwei angrenzenden Zusatzmodulen besteht. Die Zusatzmodule sind Schnittstellen zu der Compas Datenbank und zu dem Berwerbersystem für das Cockpit (vgl. Berger, 2016, S.19).

1.6 Bisherige Vorgehensweise der Kapazitätsplanung für die Kabinenbesatzung

Bisher gibt es für die Kapazitätsplanung der Kabine noch kein wirkliches Tool. Zwei Angestellte der Lufthansa Passage planen die Kapazitäten mithilfe von Excel-Tabellen. Dafür verknüpfen sie ca. 220 Tabellen miteinander (vgl. Berg u. a., 2017). Da es kaum möglich ist, jede einzelne Person konkret einzuteilen, wird jede einer Kleingruppe (KG) zugewiesen. Diese werden ähnlich wie die PUs durch Funktion, Homebase, Unterfunktion, Fluggesellschaft und ein oder mehreren Flugzeugtypen definiert. Auf Grundlage dieser KGs wird die Kapazitätsplanung durchgeführt und nur bei Bedarf können einzelne Personen genauer betrachtet werden.

Nach M2P Consulting sei „die Handlungsfähigkeit der Kapazitätsplanung in Bezug auf die zukünftige Herausforderungen stark eingeschränkt“ (M2P, 2016, S.5) und „deckt keine der definierten Soll-Funktionalitäten der Kapazitätsplanung ausreichend ab“ (M2P, 2016, S.6). Bei den Soll-Funktionalitäten handelt es sich nach M2P um langfristige Bereederung und Budgetplanung und um mittelfristige Bereederung (vgl. M2P, 2016, S.6).

2 Problemanalyse

2.1 Erhebung der Pilotendaten von der Crew Database

Im Vergleich zu COC, welches mit Daten von etwa 6.000 Piloten umgeht, muss CAB in der Lage sein, Daten von etwa 20.000 Personen ohne große Schwierigkeiten verarbeiten zu können (vgl. Wagner u. a., 2017b). Das wöchentliche Formatieren dieser Daten dauert bereits in COC einige Stunden, für CAB wäre es also fast das Vierfache. Es besteht dabei jedoch das Risiko, dass die Laufzeit sogar exponentiell ansteigt. Bei einer langen Laufzeit erhöht sich zudem auch die Fehleranfälligkeit.
Hinzu kommt, dass die Datenbank nicht lange von CAB ausgelastet werden darf, weil auch andere Schnittstellen darauf zugreifen. Eine zu starke Auslastung kann dabei zu großen Problemen in unterschiedlichen Bereichen und Systemen bei der Lufthansa Passage führen (vgl. Wagner u. a., 2017b).

Bei der Erhebung der Daten in COC werden von der Crew Database (CDB), der zentralen Datenbank, werden alle benötigten Daten aus unterschiedlichen Systemen herausgesucht und daraus neue Views erstellt, sodass sie alle tagesgenau für die nächsten 450 Tage vorliegen. Dieses Erstellen der Views wird in dieser Arbeit auch Datenaufbereitung genannt. Daraus entstehen drei Tabellen, die eine enthält Bestandsdaten für jede Person (personenbasiert), die andere enthält diese für jede PU (kumuliert) und die dritte beinhaltet Stammdaten jedes Mitarbeiters, die für die Kapazitätsplanung benötigt werden. Die hohe Anzahl an Datensätzen führt dazu, dass die Tabellen bis zu drei Millionen Einträge enthalten, die alle von der CDB erhoben und berechnet werden (vgl. Hentschel u. a., 2017).

Aus diesem Problem resultiert, dass zur Datenerhebung aus der CDB eine neue Architektur benötigt wird, um die fast 20 Jahre alte von COC zu ersetzen (vgl. Hentschel u. a., 2017). Für eine Veränderung des Architekturansatzes zur Datenaufbereitung wird ein grobes Verständnis der bisherigen Architektur von COC benötigt.

2.2 Integration in das Compas-Umfeld

Sobald CAB weit genug entwickelt worden ist, soll es, den aktuellen Anforderungen nach, in das bereits bestehende COC integriert werden. Der Kunde möchte momentan kein wei-

teres Programm haben. Im Gegensatz dazu soll aber auch ein modernes Programm entwickelt werden. Deshalb ist es unklar, ob man dieses im Endeffekt in COC integriert oder doch ein neues Programm entwickelt, was sogar als Vorlage für eine Modernisierung von COC genutzt werden kann. Das Ergebnis dieses Projektes soll auch als Entscheidungshilfe dienen.

2.3 Erfassen der Anforderungen

Bei der Anforderungsermittlung geht es darum, die passenden Anforderungen von den Stakeholdern zu ermitteln. In diesem Projekt werden die Anforderungen größtenteils von den Kabinenplanerinnen gestellt, dem sogenannten Fachbereich.

Herausforderungen bei der Ermittlung sind unterschiedliche und häufig wechselnde Anforderungen. Die Stakeholder können sie oft selber nicht genau benennen und ausdrücken, sodass die Wünsche sich zu widersprechen scheinen oder durch neue Ideen und Vorschläge ändern. Ein anderer problematischer Teil ist das Sprachverständnis während der Anforderungsermittlung. Der „Fachbereich [kennt] in den seltensten Fällen die Fachbegriffe des Entwicklers [.] und umgekehrt [...]. Somit ist eine Art 'Übersetzungsprozess' zwischen der Sprache des Fachbereichs und der des Entwicklers notwendig“ (Alpar u. a., 2016, S.319) . Für den Auftragnehmer bedeutet das eine ständige Hinterfragung aller Details und Fachbegriffe sowie die Absprache jeder Kleinigkeit mit den Stakeholdern, sodass sie vollständig ihren Anforderungen entsprechen.

2.4 Mehrfachqualifikationen der Kabinenbesatzung

Jede Person der Kabinenbesatzung kann mehreren Flotten zugeordnet sein. Eine Person wird zwar genau einer KG zugeteilt, aber diese KG kann bis zu drei, aber auch weniger Flugzeugtypen haben, für die die zugeordneten Flugbegleiter und -begleiterinnen qualifiziert sind (vgl. Berg u. a., 2017).
Auch in COC ist die Zuteilung jeder Person zu einer PU eindeutig. Jede PU wird aber, im Gegensatz zu der Kabinenbesatzung, durch einen eindeutigen Flugzeugtypen definiert. Da die PUs Basis für die Berechnungen sind, basiert COC auf der Tatsache, dass eine Person,

anders als die Kabinenbesatzung, in genau einer Rolle auf genau einer Flotte fliegt. Deshalb muss der Prototyp sinnvoll mit den Mehrfachqualifikationen der Kabinenbesatzung umgehen können.

3 Grundlagen/Methodischer Ansatz

3.1 Der Software-Lebenszyklus

Software Projekte lassen sich in einzelne Phasen unterteilen. Diese „Phasen, die ein Softwareprodukt bei seiner Herstellung und dem späteren Einsatz durchläuft“ (Brich u. Hasenbalg, 2013, S.173) nennt man den Software-Lebenszyklus eines Produktes.

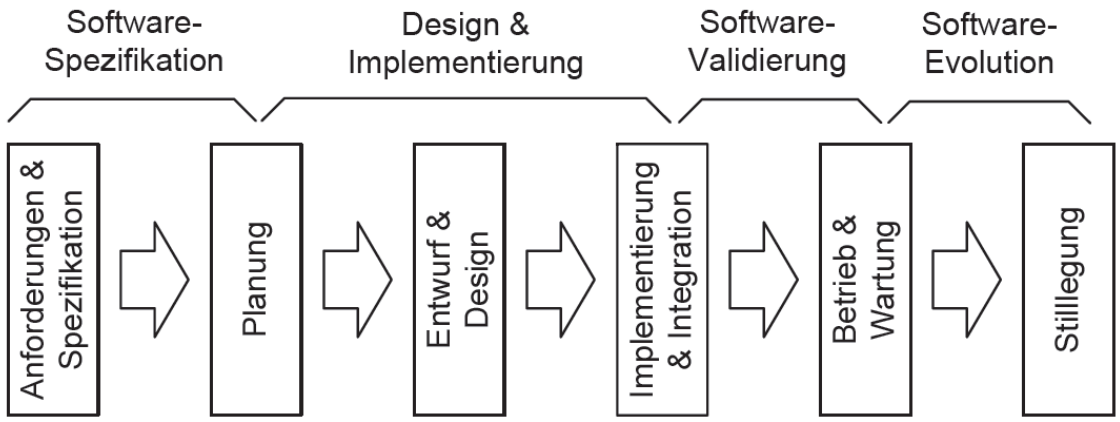

Abbildung 1: Lebenszyklus eines Softwareprojekts
(Schatten u. a., 2010, S.13)

In Abbildung 1 lassen sich die wesentlichen Schritte des Software-Lebenszyklus erkennen. Diese sind hier entweder grob in vier Stufen oder differenzierter in sechs Stufen unterteilt. Es gibt keine Festlegung über die Anzahl oder die Bezeichnungen der Abstufungen, sodass es sehr viele unterschiedliche Versionen des Software-Lebenszyklus und damit Vorgehensmodelle für die Softwareentwicklung gibt. Grundlegende Dinge, die hier durch *Software-Spezifikation*, *Design und Implementierung*, *Software-Validierung* und S*oftware-Evolution* dargestellt werden, „finden sich [jedoch] nahezu in allen Projekten wieder“ (Schatten u. a., 2010, S.13).
Detaillierter lässt sich der Software-Lebenszyklus in die sechs Phasen *Anforderung und Spezifikation*, *Planung*, *Entwurf und Design*, *Implementierung und Integration*, *Betrieb und Wartung* und *Stilllegung* aufteilen.
Die Einteilung eines Softwareprojekts in Phasen „kann den Entwicklerteams als Leitlinie dienen, um ein Software-Projekt im Team erfolgreich zu strukturieren und abzuwickeln“ (Schatten u. a., 2010, S.11). Dabei können die Phasen je nach Vorgehensmodell variieren oder wahrgenommen werden.

3.2 Vorgehen bei der Beschreibung und Analyse der Anforderungen

Die Beschreibung und Analyse von Anforderungen ist der erste Schritt bei fast jedem IT-Projekt. Zunächst wird „aus der in einer systematischen Ermittlung gewonnenen Information [.] bei der Dokumentation eine präzise Anforderungsspezifikation erstellt“ (Partsch, 2010, S.44). In diesem Fall wird in Zusammenarbeit mit dem Kunden eine Anwenderdokumentation erstellt, die die Anforderungen beschreibt. Das Ziel dabei ist, „möglichst vollständige Kundenanforderungen in guter Qualität zu dokumentieren und dabei Fehler möglichst frühzeitig zu erkennen und zu beheben“ (Pohl u. Rupp, 2015, S.11). Für so eine Anforderungsanalyse gibt es unterschiedliche Methoden, die jedoch alle „vor allem gesunden Menschenverstand voraus[setzen]“ (Partsch, 2010, S.58).
Quelle zur Ermittlung der Anforderungen sind Dokumente, bereits existierende Systeme und hauptsächlich die sog. Stakeholder. Diese sind „Person[en] oder Organisation[en], die (direkt oder indirekt) Einfluss auf die Anforderungen ha[ben]“(Pohl u. Rupp, 2015, S.21). Gemeint sind damit also alle Menschen, die in irgendeiner Weise mit der Software zu tun haben oder haben werden, z.B. der Kunde, der Nutzer, die Entwickler etc.
Da der Auftraggeber oft nicht in der Lage ist genau auszudrücken, was er will und braucht, ist es die Aufgabe der Entwickler „in Software-Projekten frühzeitig mit dem Kunden und den späteren Nutzern [zu] reden, um zu erfahren, was sie sich vorstellen“ (Kleuker, 2016, S.15).
Es muss also eine ständige Kommunikation und Zusammenarbeit sichergestellt werden. So eine Zusammenarbeit setzt „ein gewisses Verständnis der Arbeitsabläufe des Kunden, genauer dessen Geschäftsprozesse (vgl. Gadatsch, 2010), [voraus], die mit der zu entwickelnden Software im Zusammenhang stehen“ (Kleuker, 2016, S.15).

3.3 Qualitätssicherung der Anforderungen

„Grundsätzlich gilt, dass die Produkte aller Tätigkeiten bei der Softwareentwicklung [...] qualitätsgesichert [...] werden müssen“ (Winter, 1999, S.55). Deshalb „ist es notwendig, die Qualität der entwickelnden Anforderungen zu überprüfen“ (Pohl u. Rupp, 2015, S.95), um Probleme der Anforderungsermittlung (siehe Kapitel 2.3) zu lösen.
Diese ständige Qualitätssicherung dient dazu, Fehler möglichst früh in dem Softwarelebenszyklus zu erkennen und zu beheben. Der Grund dafür ist, dass die Fehlerbehebung in der Softwareentwicklung mit steigendem Projektfortschritt mehr Aufwand verursacht

(vgl. Hußmann, 2001, S.2). Aufwand kann dabei Geld, Zeit oder auch andere zu erbringende Leistung oder Einsatz sein. Das Ziel ist daher, Fehler schon möglichst früh in dem Projektverlauf zu beseitigen.

Dieser Aufwand der Fehlerbehebung ist der Grund für die Qualitätssicherung der Anforderungsermittlung und dadurch ist sie so wichtig für den Erfolg eines Projektes. Diese wichtige Rolle wird in dem folgenden Balkendiagramm dargestellt:

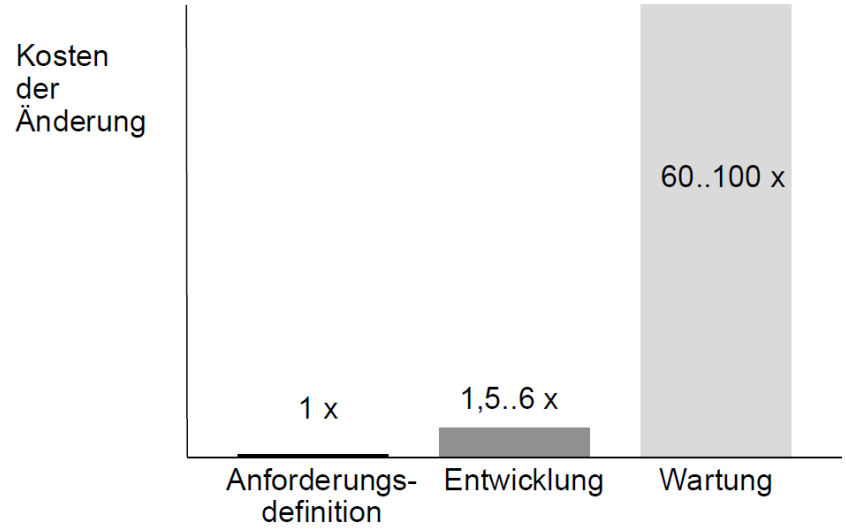

Abbildung 2: Aufwand der Fehlerbehebung in Softwareprojekten
(Hußmann, 2001, S.2)

Das Diagramm beschreibt den Aufwand der Fehlerbehebung in einem Softwareprojekt in Abhängigkeit von dem Projektfortschritt. Der Fortschritt wird dabei als Phasen des Softwarelebenszyklus dargestellt, wobei dieser vereinfacht nur in Anforderungsdefinition, Entwicklung und Wartung aufgeteilt ist.

Wird ein Fehler in der Phase der Anforderungsermittlung oder am Anfang des Projektes behoben, wird der Aufwand dafür mit dem Faktor Eins multipliziert. Sobald ein Fehler gefunden wurde, gilt es die Anforderung zu überarbeiten und ihn einfach zu korrigieren. Während der Entwicklung des Projektes ist der Aufwand meist um das 1,5 bis 6-fache höher. Grund dafür ist, dass Fehler, die sich dann noch in der Software befinden, oft mehrere Bereiche betreffen und viel verändert und berücksichtigt werden muss bevor man sie entgültig beheben kann.
Am meisten Aufwand verursacht die Fehlerbehebung, falls der Fehler erst am Ende des

Software-Lebenszyklus entdeckt wird. Dabei ist die Software schon in Betrieb und befindet sich in der Wartung. Um dort einen Fehler zu beheben, müssen oft ganze Programmabschnitte verändert werden, damit die Software wieder fehlerfrei läuft. Es hat also große Folgen, wenn ein Teil des Programmes während der Wartung verändert werden muss, weshalb der Aufwand etwa 60 bis 100 mal höher ist als zu Anfang des Projekts.
Deshalb ist es so wichtig, die einzelnen Phasen eines Software-Lebenszyklus sorgfältig zu bearbeiten und zu überprüfen, sodass keine Fehler auftreten oder diese schon so früh wie möglich erkannt werden. Es sollte daher nicht zu früh und unvorbereitet mit der Entwicklung begonnen werden.
Obwohl es Ausnahmen gibt, lohnt sich der Aufwand in der Analyse meistens, da „viele schwerwiegende Fehler in den frühen Phasen IS-Entwicklung [Informationssystem-Entwicklung; Anmerk. d. Verf.] gemacht werden“ (Alpar u. a., 2016, S.316). Trotzdem wird in vielen Fällen zu voreilig mit der Entwicklung angefangen.
"Die Analyse der in einer Anforderungsdefinition festgelegten Anforderungen zielt letztlich darauf ab, Aufschluss über die Qualität der Anforderungsbeschreibung zu erhalten“ (Partsch, 2010, S.51). Kriterien für die Qualität lassen sich hauptsächlich in Inhalt, Dokumentation und Abgestimmtheit gliedern. Es wird sich also damit befasst, ob die Anforderungen vollständig, detailliert, passend dokumentiert und mit allen Stakeholdern abgestimmt sind. Für jeden der drei Qualitätsaspekte gibt es damit unterschiedliche Prüfkriterien (vgl. Pohl u. Rupp, 2015, S.97).

3.3.1 Qualitätsaspekt Inhalt

„Der Qualitätsaspekt »Inhalt« bezieht sich auf die Überprüfung von Anforderungen auf inhaltliche Fehler“ (Pohl u. Rupp, 2015, S.98). Dafür gibt es acht Prüfkriterien(vgl. Pohl u. Rupp, 2015, S.98):

Vollständigkeit: Wurden alle relevanten Anforderungen erfasst?

Korrektheit: Beschreibt jede Anforderung die dafür notwendigen Informationen?

Verfolgbarkeit: Können die Anforderungen verfolgt, also z.B. auf die Quelle zurückgeführt werden?

Adäquatheit: Enthalten die Anforderungen die Bedürfnisse und Wünsche der Stakeholder angemessen?

Konsistenz: Gibt es Widersprüche zwischen den Anforderungen?

Vorzeitige Entwurfsentscheidungen: Wurden Entwurfentscheidungen vorweggenommen, die nicht durch Randbedingungen bestimmt sind?

Überprüfbarkeit: Können Abnahme- und Prüfkriterien anhand der Anforderungen definiert werden?

Notwendigkeit: Trägt jede Anforderung zu dem definierten Ziel bei?

3.3.2 Qualitätsaspekt Dokumentation

Bei der Dokumentation geht es darum, die „Anforderungen auf Mängel in der Dokumentation bzw. auf Verstöße gegen geltende Dokumentationsvorschriften“(Pohl u. Rupp, 2015, S.99) zu überprüfen. Hierbei gibt es folgende vier Prüfkriterien(vgl. Pohl u. Rupp, 2015, S.99f.):

Konformität: Wurden die Anforderungen in dem vorgeschriebenen Dokumentationsformat strukturiert und in der richtigen Modellierungssprache dokumentiert?

Verständlichkeit: Können die Anforderungen in dem gegebenen Kontext ggf. mithilfe eines Glossars verstanden werden?

Eindeutigkeit: Ist eine eindeutige Interpretation möglich?

Konfirmität mit Dokumentationsregeln: Sind vorgegebene Dokumentationsregeln und -richtlinien eingehalten worden?

3.3.3 Qualitätsaspekt Abgestimmtheit

Die Abgestimmtheit stellt sicher, dass keine “Mängel in der Abstimmung der Anforderungen unter relevanten Stakeholdern“(Pohl u. Rupp, 2015, S.100) vorliegen. Auch dafür gibt es folgende drei Prüfkriterien(vgl. Pohl u. Rupp, 2015, S.100) :

Abstimmung: Wurde jede Anforderung mit relevanten Stakeholdern abgestimmt?

Abstimmung nach Änderungen: Wurde jede Änderung der Anforderungen auch abgestimmt?

Konflikte: Wurden alle bekannten Konflikte gelöst?

3.4 Anforderungskategorisierung

Anforderungen werden kategorisiert und nach Wichtigkeit eingeteilt. Das ist hilfreich, da die Anforderungen unterschiedlich zur Zufriedenheit der Stakeholder beitragen (vgl.

Pohl u. Rupp, 2015, S.24). Durch Kategorisierung lassen sie sich leichter einordnen, um Aufwand und Priorität besser abschätzen zu können.
In diesem Fall werden die Anforderungen nach dem Kano-Modell kategorisiert. Demnach gibt es drei Kategorien (Pohl u. Rupp, 2015, vgl. S.24):

Basisfaktoren sind selbstverständliche unterbewusste Systemmerkmale, die vorausgesetzt werden.

Leistungsfaktoren sind bewusste, explizit geforderte Systemmerkmale.

Begeisterungsfaktoren sind für den Stakeholder unbekannte Systemmerkmale, die er während der Benutzung als angenehme Überraschung entdeckt.

Mit der Zeit werden aus Begeisterungsfaktoren Leistungsfaktoren und aus Leistungsfaktoren Basisfaktoren, da der Nutzer sich an die Merkmale gewöhnt und sie irgendwann voraussetzt. Das Modell lässt sich grafisch darstellen:

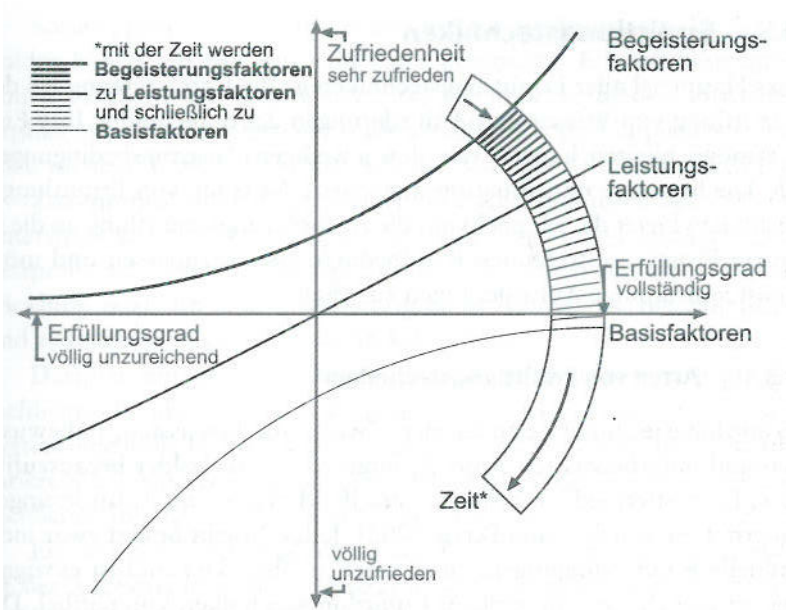

Abbildung 3: Grafische Darstellung des Kano-Modells
(Pohl u. Rupp, 2015, S.25)

Die Grafik zeigt die Zufriedenheit der Stakeholder in Abhängigkeit von dem Erfüllungsgrad der jeweiligen Faktoren. Auf einer dritten Achse wird die Zeit dargestellt, wodurch die Anforderungen die Kategorien wechseln können.
Die Basisfaktoren müssen erfüllt sein, um eine massive Unzufriedenheit der Stakeholder zu vermeiden (Zhu u. a., 2010, vgl. S.106). Sie werden vorrausgesetzt und haben daher

nur den Nutzen, Unzufriedenheit zu vermeiden. Daher verläuft die Funtkion negativ exponentiell.
Die Funktion der Leistungsfaktoren verläuft linear und schneidet den Nullpunkt. Diese Faktoren erzeugen proportional Unzufriedenheit, wenn sie nicht erfüllt sind, und Zufriedenheit, wenn sie erfüllt sind (Zhu u. a., 2010, vgl. S.106).
Da die Begeisterungsfaktoren vorher den Stakeholdern nicht bekannt sind, kann ihr unzureichender Erfüllungsgrad keine Unzufriedenheit verursachen, da sie nicht erwartet werden oder bekannt sind (Zhu u. a., 2010, vgl. S.106). Das Vorhandensein dieser Faktoren freut jedoch die Stakeholder, wodurch es zu einem überproportionalen Nutzen bei steigender Erfüllung kommt. Deshalb ist die Funktion der Begeisterungsfaktoren linear und erreicht niemals den Wert null oder weniger, aber steigt sehr schnell an.

Es ist ungeklärt, ob man diese Faktoren und ihre Eigenschaften so auf die Anforderungen bei der Erstellung eines Prototyps anwenden kann. Laut Pombergers Definition soll ein explorativer Prototyp ermöglichen, die Vorstellungen der Anwender „anhand von Anwendungsbeispielen zu prüfen und die gewünschte Funktionalität zu ermitteln“ (Pomberger u. Pree, 2004, S.27), es soll also die Analyse und die Anforderungsspezifikation unterstützen. Das bedeutet, dass der Prototyp nicht alle Anforderungen enthalten kann, besonders die Leistungs- und Begeisterungsfaktoren nicht, und genau im Gegenteil dazu dient, Anforderungen zu ermitteln.
Kanos Begeisterungsfaktoren widersprechen zudem dem weit verbreiteten Vorgehen bei einer erfolgreichen Anforderungsanalyse. Wie von Pohl und Rupp beschrieben ist dafür „eine gute Kommunikation und [...] Qualität der Zusammenarbeit mit den Stakeholdern“ (Pohl u. Rupp, 2015, S.33) notwendig, um ihn „erfolgreich in den Ermittlungsprozess einzubinden“ (Pohl u. Rupp, 2015, S.34). Durch diese enge Zusammenarbeit werden alle Anforderungen abgesprochen, sodass es theoretisch kaum noch Begeisterungsfaktoren geben kann.

3.5 Abgrenzung des Systems und Systemkontextes

Die Abgrenzung eines Systemes zu seiner umgebung und zu den Schnittstellen ist essentiell für den Entwurf eines korrekten Produktes. Dafür müssen die System- und Kontextgrenzen bestimmt werden.
Dadurch wird Aufwand vermieden, indem bereits vorhandene Funktionen evtl. integriert

oder nur in etwas abgeänderter Weise übernommen werden. Außerdem bringt sie Klarheit über die Arten der Informationsgewinnung und -weiterverarbeitung, um Missverständnisse zu vermeiden. Es ist wichtig, „die Grenzen des Systems zum Systemkontext und die Grenzen des Systemkontexts zur irrelevanten Umgebung zu bestimmen“ (Pohl u. Rupp, 2015, S.20), da der Systemkontext „auch die Anforderungen an das zu entwickelnde System bestimmt“ (Pohl u. Rupp, 2015, S.20).
Der praktische Teil zu diesem Kapitel fällt weg, da diese Arbeit COC als Grundlage nutzt und Systemgrenzen und -schnittstellen damit schon klar definiert sind. Nach der Erarbeitung des Lösungsentwurfs ist es dann die Entscheidung des Kunden, ob CAB in COC eingegliedert werden soll und die Abgrenzung damit wegfällt oder ob System und Systemkontext für ein neues Programm neu bestimmt werden müssen.

3.6 Arten und Ziele des Prototypings

Prototyping beschreibt „eine Vorgehensweise bei der Softwareentwicklung, bei der nicht sofort ein endgültiges Softwaresystem, sondern zunächst ein oder mehrere Prototypen erstellt werden“ (Brich u. Hasenbalg, 2013, S.152). Dabei wird nach Kuhrmann zwischen horizontalem und vertikalem Prototyping unterschieden (vgl. Kuhrmann, 2012):
Horizontales Prototyping bezieht sich auf einen bestimmten Bereich der Software, z.B. die Benutzeroberfläche. Der Bezug zu der technischen Funktionalität und der tatsächlichen Implementierung ist dabei nicht gegeben. Nur die eine Ebene des Programms soll vorgestellt werden.
Beim **vertikalen Prototyping** wird ein Ausschnitt komplett in allen Ebenen vollständig implementiert. Diese Art dient zur Demonstration von komplexer Funktionalität.

Darüber hinaus unterscheidet die IEEE das Prototyping anhand von Anwendungszwecken (vgl. Lichter u. a., 1994, S.826):

Exploratives Prototyping wird benutzt, wenn man das Problem und die Anforderungen noch nicht genau kennt. Indem viele Ideen und Ansätze ausprobiert werden, lernt der Entwickler die Arbeit und die Anforderungen des Kunden besser kennen, um die Anforderungen zu identifizieren.

Experimentelles Prototyping zielt auf das Sammeln von Erfahrungen und Ideen. Der Anwender kann mit dem Prototypen experimentieren, um Ideen an das Programm

weiterzuentwickeln und Anforderungen zu konkretisieren. Währenddessen enthält der Entwickler Eindrücke von der Realisierbarkeit des Systems und technischen Herausforderungen.

Evolutionäres Prototyping stellt die Softwareentwicklung nicht als ein temporäres Projekt, sondern als eine fortlaufende Entwicklung dar. Dabei wird das Programm nach und nach erweitert, wobei der Entwickler eng an der Seite des Anwenders das System immer weiter erweitert.

Die Definition des evolutionären Prototypings widerspricht jedoch der allgemeinen Definition nach Brich u. Hasenbalg (2013). Das Ergebnis des explorativen und des experimentellen Prototypings ist ein Prototyp im engeren Sinne, der der Demonstration dient. Dieser wird nach der Erfüllung seiner Aufgaben nicht mehr benötigt (vgl. Liggesmeyer, 2012, S.21). Beim evolutionären Prototyping hingegen entsteht ein Pilotsystem, das der Kern des Produktes ist. Der Prototyp wird also fließend zum laufenden Produkt (vgl. Liggesmeyer, 2012, S.24) und ist damit eigentlich kein Prototyp laut Definition mehr, sondern schon ein endgültiges Softwaresystem.

4 Istanalyse

4.1 Bestandsrechnung in Compas Cockpit

Der aktuelle Algorithmus für die Bestandsrechnung in COC ermittelt zuerst den Bruttobestand der Besatzungen (vgl. DLH, 1995, S.8). Die Einheiten für alle Bestands- und Bedarfsrechnungen sind Beschäftigungstage (BT) oder Stunden.
Der Bestand wird ermittelt, indem der aktuelle Pilotenbestand mit allen Formen von Zu- und Abgängen z.B. durch Umschulungen, Altersabgänge, Neueinstellungen oder Kündigungen sowie allen bezahlungswirksamen Fehlzeiten wie Teilzeit, Mutterschutz oder Fluguntauglichkeit errechnet wird (vgl. LSY, 2001, S.19). Insgesamt gibt es 16 Kategorien, die hier aber der Übersicht halber nicht alle genannt werden.
Von diesem Bruttobestand werden alle weiteren Fehlzeiten, die sich grob in Urlaub und Krankheit einteilen lassen, subtrahiert, was zu dem Nettobestand führt (vgl. DLH, 1995, S.8).
Das Ergebnis aller verfügbaren Piloten ergibt sich dann durch die Subtraktion von freien Tagen, die den Piloten zustehen. Auch die Arten und die Berechnung der freien Tage werden hier nicht weiter erläutert.
Dadurch ergibt sich der Bestand aller verfügbaren Piloten in BTs. Diese BTs lassen sich auf Tages-, Wochen- und Monatsbasis darstellen und zusammenfassen.

4.2 Anforderungsanalyse- und kategorisierung

Die Planerinnen formulierten folgende Anforderungen (vgl. Berg u. a., 2017):

Basisfaktoren: Es soll ein Programm erstellt werden, dass die Kapazitäts- und später auch die Schulungsplanung automatisch regelt. Dafür müssen vor allem die Gruppen und Untergruppen eingeteilt und dargestellt werden können. Der Zeitraum soll 15 Monate betragen, um die Urlaubsplanung des jeweils kommenden Jahres mit einfließen lassen zu können. Eine Bestandsrechnung wird also benötigt, die später um die Bedarfsrechnung ergänzt werden soll. Wird eine Integration in das Compas-Umfeld beschlossen, wird das Programm nach und nach erweitert bis es in etwa den Funktionalitäten von COC entspricht.

Der Kunde legt außerdem viel Wert auf Nachvollziehbarkeit. Es soll bei jeder Zelle der Tabellen verstanden werden können, woher die Daten her kommen oder wie sie

berechnet werden. Dazu soll es wie bei COC die Möglichkeit geben, sich einzelne Detailabfragen anzeigen lassen zu können.

Leistungsfaktoren: Da es sich um den Entwurf eines Prototyps zur Kapazitätsplanung handelt, ist die Bedarfsrechnung und die Schulungsplanung ein Leistungsfaktor. Dazu gehören auch Bewerbungen für Schulungen. Diese Leistungsfaktoren werden aber später ergänzt, sodass sie nicht Bestandeil dieses Lösungsentwurfs sind.
Darüber hinaus wird eine gute Performance erwartet, um die Planung zu erleichtern und Zeit zu sparen.

Begeisterungsfaktoren: Modernes Design ist einer der Begeisterungsfaktoren. Dabei ist wünschenswert, das Programm so zu designen, dass sich Cockpit- und Kabinenplaner/innen ersetzen können und die Bedienung und Anordnung der Elemente gleich ist.

Da sich diese Arbeit aber nur mit der Entwicklung eines Lösungsentwurfs für den Prototypen beschäftigt, stehen die Basisfaktoren im Vordergrund. Die Leistungs- und Begeisterungsfaktoren sind in einem explorativen Prototypen noch nicht vorhanden. Sie kommen im Verlauf des Projektes dazu. Trotzdem ist eine Bestimmung davon wichtig, um das System bereits etwas abzugrenzen und Schwerpunkte festzulegen.

4.3 Die Schnittstelle: Das Crew Management System

Die Quelle, aus der COC die benötigten Daten entnimmt, ist das Crew Management System (CMS) oder genauer gesagt die CDB. Unter dem CMS wird das gesamte Planungssystem der Lufthansa verstanden, das aus vielen verschiedenen einzelnen Systemen besteht. Im Mittelpunkt davon steht die CDB, wo die Daten zusammen fließen und in einer Datenbank gespeichert werden. Es wird also von sehr vielen Seiten aus auf die CDB zugegriffen, was ihr sehr viel Leistung und Performance abverlangt.
Damit COC alle benötigten Daten in der tagesgenauer Form erhalten kann, bereitet die CDB diese Daten zwei mal im Monat auf, indem sie sie aus allen anderen Tabellen anderer Systeme herausfiltert und dann Views erstellt, die der benötigten Form entspricht. Das Erstellen dieser Views verursacht dabei sehr viel Aufwand und Auslastung der CDB.
Wenn über COC die aktuellen Daten dann angefordert werden, werden sie in die eigene Compas-Datenbank geladen, wo direkt drauf zugegriffen werden kann. Das Aktualisie-

ren der Tabellen aus der Views nimmt dabei nicht viel Zeit in Anspruch, da die Daten nur geladen und nicht verändert oder formatiert werden müssen.

5 Konkrete Modernisierung und Anpassung von Compas Cockpit

5.1 Veränderungen bei der Datenaufbereitung

Um Laufzeit- und Performanceprobleme bei der Datenaufbereitung zu vermeiden, wird eine andere Vorgehensweise als in COC benötigt. Die einfachste Möglichkeit dazu ist, alle Rohdaten, aus denen die Views erstellt werden, unverändert von der CDB in den eigenen Teil der Datenbank von COC zu laden. Von dort aus werden die Views von einem für COC dedizierten Server, der innerhalb des CMS liegt, zur Verfügung gestellt (vgl. dazu Becker u. a., 2017). Da der Server wie die CDB im CMS liegt, gilt für ihn auch der gleiche Sicherheitsstandart.
Dadurch hat die CDB nur noch die Aufgabe, die Tabellen der Rohdaten zur Verfügung zu stellen, und es wird ihr keine große Leistung abverlangt. Die Auslastung wird von der CDB auf den Compas-eigenen Server verschoben. Weil COC diesen Server nur nutzt, um einige zeitbasierte wiederkehrende Aufgaben auszuführen, ist es kein Problem, diesen für eine längere Dauer auszulasten. Selbst bei starker Auslastung wird damit kein anderes System beeinträchtigt. Durch eine höhere Programmiersprache und einem effizienteren Algorithmus kann die Datenaufbereitung außerdem deutlich schneller erfolgen. Nach der Erstellung der Views werden alle daraus erhaltenen Daten wie bislang wieder in den Compas-eigenen Teil der CDB geladen, worauf das Programm darauf zugreift.

Bei der Datenaufbereitung wird mit personenbezogenen Daten gearbeitet, weshalb beim Erstellen der Views „die Bestimmungen des Bundesdatenschutzgesetzes“ (Maier, 2005, S.6) und weitere „spezielle Schutzvorschriften [,die] zwischen den Geschäftsleitungen und den Mitarbeitervertretungen [der Lufthansa AG; Anmerk. d. Verf.] definiert [wurden]“ (Maier, 2005, S.6), gelten. Obwohl für beide Systeme die gleichen Sicherheitsvorkehrungen gelten, musste diese Vorgehensweise diskutiert und eine Genehmigung benötigt werden.

5.2 Veränderungen in der Bestandsrechnung

In der in Kapitel 4.1 beschriebenen Bestandsrechnung von COC müssen einige Zeilen angepasst werden, damit sie der bisherigen Bestandsrechnung der Kabine entspricht. Diese

Anpassung wurde mit dem Kunden genau besprochen (vgl. Winkel u. a., 2017).
Einige Zeilen wie „Verlängerer Teilzeit“ oder „Alterssonderurlaub“ fallen weg, da so etwas bei der Kabinenbesatzung nicht vorkommen kann. Eine Änderung wird für das Feld „Umschulungen“ benötigt, die in COC einfach verrechnet wurden. In CAB ist es notwendig, zwischen Schulungen, die kürzer als fünf Tage, und welchen, die mindestens fünf Tage dauern zu unterscheiden, weshalb zwei Zeilen für die beiden Arten der Schulungen benötigt werden.
Bei der restlichen Bestandsrechnung kann genau so vorgegangen werden, wie es bereits in COC für die Cockpit-Mitarbeiter implementiert ist.

5.3 Abbilden der Mehrfachqualifikationen und Einbindung der Kleingruppen

Nach Absprache des Problems der Mehrfachqualifikationen mit dem Kunden wurde auch für dieses Problem eine Lösung gefunden (vgl. Wagner u. a., 2017a):
Den Anforderungen nach, soll es möglich sein, den PUs manuell die KGs zuzuordnen. Die KGs müssen vom Nutzer vorher genannt werden und können nicht von ihm selbst bearbeitet werden. Es soll für ihn aber möglich sein, PUs anzulegen, zu bearbeiten und zu entfernen. Jede KG wird dann genau einer PU vom Nutzer zugeordnet.
Deshalb muss anstatt einer direkten Verbindung zwischen einer Person und einer PU ein Zwischenschritt mit den KGs eingebaut werden. Jede Person wird also anstatt einer PU mit einer KG verknüpft. Danach ist eine neue Tabelle notwendig, in der die KGs den PUs zugeordnet werden, die vom Anwender bearbeitet werden kann. Die Mehrfachqualifikationen fließen somit in das Konzept der PUs ein und es kann für jede PU der Bestand ermittelt werden. Eine eindeutige Bedarfsrechnung ist mit dieser Methode aber noch nicht möglich.

6 Zusammenfassung und Ausblick

6.1 Änderungen an Compas Cockpit um es für eine automatisierte Kapazitätsplanung der Kabinenbesatzung zu nutzen

Um auf Grundlage von COC die Kapazitätsplanung der Kabinenbesatzung zu automatisieren, müssten einige Dinge des Programms angepasst werden:
Die Funktionsweise der Datenaufbereitung muss aufgrund der hohen Anzahl an Daten verändert werden. Die beste und einfachste Lösung dafür ist, das Erstellen der Views von der CDB auf einen Server zu verlegen, der nur von COC genutzt wird. Dazu kann der Algorithmus durch eine höhere Programmiersprache wie z.B. Python verbessert werden.
Um Mehrfachqualifikationen der Kabinenbesatzung für Flotten sinnvoll in das Programm mit aufzunehmen, werden die KGs, die die Qualifikationen enthalten vom Nutzer selbst PUs zugewiesen. Dadurch ist eine eindeutige Bestandsrechnung möglich. Die Bestandsrechnung an sich ist für die Kapazitätsplanung der Cockpit- und der Kabinenbesatzung nahezu dieselbe.
Desweiteren ist es nötig, viele Kleinigkeiten, z.B. Datenfelder oder Prämissen für eine Prognosenberechnung, anzupassen und sich mit dem Kunden alle Details anzuschauen. Daraus ergibt sich, welche Funktionen er nicht mehr oder zusätzlich benötigt. Die hauptsächlichen Funktionen bleiben aber im Vergleich zu COC unverändert.
Durch die Ähnlichkeit der Kapazitätsplanung in beiden Bereichen, ist es relativ einfach und vorteilhaft, CAB in COC einzugliedern und es als Vorlage zu nutzen.

6.2 Probleme bei der Anpassung der Bedarfsrechnung der Kabinenbesatzung nach dem Vorbild von Compas Cockpit

Nachdem die Bestandsrechnung relativ einfach angepasst werden kann, um de Bestand des Kabinenpersonals zu ermitteln, entstehen bei der Anpassung der Bedarfsrechnung weitere Probleme. Der Bedarf wird als nächster Schritt benötigt, um aus Bestand und Bedarf die Kapazität zu ermitteln. Bekommt die Lufthansa z.B. ein neues Flugzeug, einen Airbus A340, wird für dieses Flugzeug eine Cockpitbesatzung benötigt. Dieser Bedarf fällt auf eindeutige PUs zurück, die genau die Piloten beinhalten, die in den benötigten Funktionen eine A340 fliegen. Da jede PU des Cockpits eindeutig ist, kann der Bedarf einfach zugeordnet werden.

Aufgrund der Mehrfachqualifikationen entstehen bei der Bedarfszuordnung der Kabinenbesatzung aber Schwierigkeiten. Für die Kabinenbesatzung gibt es für eine Rolle z.B. eine PU1, in denen die Leute auf einer A380 und einer A340 fliegen, aber auch eine PU2, in denen sie auf A340 und Boeing 747 eingeteilt sind. Entsteht nun ein Bedarf an Kabinenpersonal für eine A340 ist es unklar, ob dieser der PU1 oder der PU2 zugeschrieben wird. Ein erster Lösungsansatz wäre es, den Bedarf prozentual auf alle möglichen PUs zu verteilen und diese zu gewichten. Wie genau damit umgegangen wird, muss aber mit dem Kunden ausführlich diskutiert werden, damit man sich auf eine Lösung einig wird.

6.3 Compas Cockpit als Vorbild zur automatisierten Kapazitätsplanung der Kabinenbesatzung

Kurzfristig gesehen ist die Integration in COC eine Lösung, mit der man mit wenig Aufwand das gewünschte Ziel, eine automatisierte Kapazitätsplanung der Kabinenbesatzung, erreichen kann.
Meiner Meinung nach ist diese Methode jedoch kaum nachhaltig. COC ist technisch gesehen auf einem fast 20 Jahre alten Stand. Gerade in der IT-Branche, die sich sehr schnell weiterentwickelt, ist dies eine sehr lange Zeit. Für die Gegenwart reicht es bis jetzt noch, aber in einigen Jahren können sich die Anforderungen an ein modernes, komplexes Programm und seinen Funktionen so stark verändern, dass das alte COC dafür nicht mehr ausreicht. Deshalb wäre es Verschwendung, den einfachen Weg zu gehen, um in einigen Jahren trotzdem ein fast komplett neues und modernes Programm zu entwickeln, wenn festgestellt wird, dass die Grenzen von CAB überschreitet werden.
Es sollte lieber jetzt Aufwand in ein neues, eigenständiges und modernes Programm investiert werden; früher oder später reicht die bisherige Lösung nicht mehr aus und doppelte Arbeit bleibt dadurch erspart. Außerdem könnte CAB vielleicht auch ein Vorbild für eine Modernisierung von COC sein. Es ist im Sinne der Lufthansa, langfristig die beste Lösung in allen Bereichen zu finden, um auch in Zukunft ein einflussreiches und erfolgreiches Unternehmen zu bleiben.

6.4 Weitere Vorgehensweise

Zunächst müssen Probleme und Unterschiede bei der Bedarfsrechnung diskutiert und geklärt werden. Sobald es eine Lösung gibt, diese anzupassen, kann auf Grundlage dar-

auf die Deltarechnung erstellt werden. Das entspricht dem gröbsten Teil der Kapazitätsplanung. Hinzu kommen noch Detailabfragen und Prämissen, die abgesprochen werden müssen, bevor die Kapazitätsplanung nach Vorbild von COC in einem Prototypen implementiert werden kann. Als nächstes wäre es sinnvoll, eine automatisierte Lösung für die Schulungsplanung der Kabinenbesatzung zu entwickeln.

Der Prototyp dient als Entscheidungshilfe, um die wahrscheinlich wichtigste Anforderung zu klären. Die Stakeholder können sich damit vor Augen führen, wie gut COC als Vorbild für die Kapazitätsplanung der Kabinenbesatzung geeignet ist. Sie müssen dann entscheiden, ob sie sie in COC integrieren oder in ein größeres Projekt investieren und eine komplett neue Software zu entwickeln.

Literatur

[Alpar u. a. 2016] ALPAR, Paul ; ALT, Rainer ; BENSBERG, Frank ; GROB, Heinz L. ; WEIMANN, Peter ; WINTER, Robert: *Anwendungsorientierte Wirtschaftsinformatik : Strategische Planung, Entwicklung und Nutzung von Informationssystemen*. 8.Auflage. Wiesbaden : Springer Vieweg, 2016 (477152740). – ISBN 978–3–658–14146–2

[Becker u. a. 2017] BECKER, Torsten ; BERGER, Iwan ; GARSKE, Julian: *Gespräch über Veränderungen bei der Datenaufbereitung*. Raunheim, 26. Juni 2017

[Berg u. a. 2017] BERG, Stefanie ; WAGNER, Vanessa J. ; GARSKE, Julian: *Erstes Gespräch mit den Planerinnen über die aktuelle Vorgehensweise*. Frankfurt am Main, 24. Mai 2017

[Berger 2016] BERGER, Iwan: *Anwendungsdokumentation gemäß CMS-BetrVbg; COMPAS - Kapazitätsplanung Cockpit*. Version 1.20. Raunheim: Lufthansa Systems GmbH & Co.KG, 4. Mai 2016

[Brich u. Hasenbalg 2013] BRICH, Stefanie ; HASENBALG, Claudia: *Kompakt-Lexikon Wirtschaftsinformatik : 1.500 Begriffe nachschlagen, verstehen, anwenden*. Wiesbaden : Springer Gabler, 2013. – ISBN 978–3–658–03028–5

[DLH 1995] DLH: *Funktionale Beschreibung Kapazitätsplanung Cockpit*. Version 0.1 - Entwurf. Frankfurt am Main: Deutsche Lufthansa AG, 20. März 1995

[Gadatsch 2010] GADATSCH, Andreas: *Grundkurs Geschäftsprozess-Management*. Wiesbaden : Vieweg + Teubner, 2010. – ISBN 978–3–8348–9346–8

[Hentschel u. a. 2017] HENTSCHEL, Norman ; RAPP, Dominik ; GARSKE, Julian: *Gespräch über die Datenbank-Schnittstelle*. Raunheim, 21. Juni 2017

[Hußmann 2001] HUSSMANN, Heinrich: *Anforderungsermittlung; Vorlesungsskript in „Softwaretechnologie 2“*. Dresden: Technische Universität Dresden, 2001. – URL: `http://st.inf.tu-dresden.de/Lehre/WS00-01/st2/vorlesung/st2k2a-extra.pdf`; abgerufen am 05.07.2017

[Kleuker 2016] KLEUKER, Stephan: *Grundkurs Datenbankentwicklung - Von der Anforderungsanalyse zur komplexen Datenbankanfrage*. 4. Auflage. Wiesbaden : Springer Vieweg, 2016. – ISBN 978–3–658–12338–3

[Kuhrmann 2012] KUHRMANN, Marco: Prototyping. In: *Enzyklopädie der Wirtschaftsinformatik, Online-Lexikon*. München : Hrsg.: Norbet Gronau, Jörg Becker, Elmar J. Sinz, Leena Suhl und Jan Marco Leimeister, September 2012. – URL: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/Vorgehensmodell/Prototyping/index.html; abgerufen am 26.06.2017

[Lichter u. a. 1994] LICHTER, Horst ; SCHNEIDER-HUFSCHMIDT, Matthias ; ZÜLLIGHOVEN, Heinz: Prototyping in Industrial Software Projects-Bridging the Gap Between Theory and Practice. In: *IEEE Transactions on Software Engineering* Vol. 20 (1994), S. 825–832

[Liggesmeyer 2012] LIGGESMEYER, Peter: *Software Entwicklung 2 - Prozessmodelle; Vorlesungsskript*. Kaiserslautern: Technische Universität Kaiserslautern, 2012. – URL: http://agde.cs.uni-kl.de/teaching/se2/ss2012/material/vorlesung/se2_ss2012_03_Prozessmodelle.pdf; abgerufen am 27.06.2017

[LSY 2001] LSY: *Anhang zum Benutzerhandbuch - Berechnungen*. Version 1.2. Frankfurt am Main: Lufthansa Systems GmbH & Co.KG, 26. September 2001

[M2P 2016] M2P CONSULTING (Hrsg.): *Projektergebnisse der Prüfung von M2P*. Frankfurt am Main: M2P Consulting, Februar 2016

[Maier 2005] MAIER, P.: *Security Handbuch für das dezentrale CMS (CMS/D)*. Version 9.0. Frankfurt am Main: Deutsche Lufthansa Aktiengesellschaft, 2. Mai 2005

[Partsch 2010] PARTSCH, Helmuth A.: *Requirements-Engineering systematisch : Modellbildung für softwaregestützte Systeme*. 2. überarbeitete und erweiterte Auflage. Berlin, Heidelberg : Springer-Verlag, 2010. – ISBN 978–3–642–05358–0

[Pohl u. Rupp 2015] POHL, Klaus ; RUPP, Chris: *Basiswissen Requirements Engineering*. 4. Edition. Heidelberg : dpunkt.verlag, 2015. – ISBN 978–3–89864–550–8

[Pomberger u. Pree 2004] POMBERGER, Gustav ; PREE, Wolfgang: *Software Engineering: Architektur-Design und Prozessorientierung*. 3., völlig überarbeitete Auflage. Carl Hanser Verlag GmbH & Co.KG, 2004. – ISBN 978–3–446–22788–0

[Schatten u. a. 2010] SCHATTEN, Alexander ; DEMOLSKY, Markus ; WINKLER, Dietmar ; BIFFL, Stefan ; GOSTISCHA-FRANTA, Erik ; ÖSTREICHER, Thomas: *Best Practice Software-Engineering*. Heidelberg : Spektrum Akademischer Verlag, 2010. – ISBN 978–3–8274–2487–7

[Wagner u. a. 2017a] WAGNER, Vanessa J. ; WINKEL, Alfred ; GARSKE, Julian: *Gespräch über Mehrfachqualifikationen und die Datenerhebung*. Lufthansa-Basis Frankfurt am Main, 26. Juni 2017

[Wagner u. a. 2017b] WAGNER, Wolfgang ; BECKER, Torsten ; RAPP, Dominik ; GARSKE, Julian: *Erstes Gespräch über Compas Kabine*. Raunheim, 22. Mai 2017

[Winkel 2017] WINKEL, Alfred: *High Level Items - Personalbestandsermittlung Kabine*. Version 1.3. Frankfurt am Main: Deutsche Lufthansa AG, 20. Februar 2017

[Winkel u. a. 2017] WINKEL, Alfred ; WAGNER, Vanessa J. ; GARSKE, Julian: *Gespräch über die detaillierte Bestandsrechnung und die Datenbank-Strukturen*. 4. Juli 2017. – Aufzeichnung: Meeting 05.07. - Bestandsrechnung detailliert.docx

[Winter 1999] WINTER, Mario: *Qualitätssicherung für objektorientierte Software: Anforderungsermittlung und Test gegen die Anforderungsspezifikation*, Fachbereich Informatik der FernUniversität - Gesamthochschule - in Hagen, Doktorarbeit, 1999

[Zhu u. a. 2010] ZHU, Dauw-Song ; LIN, Chih-Te ; TSAI, Chung-Hung ; WU, Ji-Fu: A Study on the Evaluation of Customers Satisfaction - the Perspective of Quality. In: *International Journal for Quality Research* (2010), August, S. 105–115

Glossar

Begriff	Erläuterung
Bestandsrechnung	Berechnung des verfügbaren Bestands in Beschäftigungstagen; dieser wird aus dem Bruttobestand und verschiedenen Ab- und Zugängen errechnet
Compas Cabin	Gefordertes Programm; soll in Compas Cockpit integriert werden und die Kapazitäts- und Schulungsplanung der Kabinencrew automatisieren
Compas Cockpit	Programm zur Kapazitäts- und Schulungsplanung der Cockpit-Mitarbeiter; für die Lufthansa von Lufthansa Systems 1999 entwickelt und seitdem ständig erweitert
Crew Database	Zentrale Datenbank der Lufthansa mit Schnittstellen zu vielen unterschiedlichen Systemen; stellt die erforderlichen Daten für Compas zusammen
Crew Management System	Überbegriff für alle Schnittstellen-Systeme der Lufthansa mit der Crew Database im Mittelpunkt
Deltarechnung	Der Bestand wird ins Verhältnis zum Bedarf gesetzt und daraus das Delta ermittelt
Funktion	auch Function; Rolle und Aufgaben, die Piloten oder die Kabinenbesatzung auf einem Flug übernehmen
Homebase	Flughafen an dem eine Person oder Gruppe stationiert ist
IEEE	Institute of Electrical and Electronics Engineers; Berufsverband von Igenieuren aus Elektro- und Informationstechnik
Kapazitätsplanung	Mithilfe der Deltarechnung sollen dabei die richtige Anzahl an Personen mit den richtigen Qualifikationen zum richtigen Zeitpunkt verfügbar sein
Kleingruppe	Gruppen für die Kap.- und Schulungsplanung der Kabinenebesatzung; definiert durch Funktion, Homebase, Unterfunktion, Fluggesellschaft und Flugzeugtyp(en)
NetLine/Crew	Von Lufthansa Systems entwickeltes Programm, das alle Phasen des Crewmanagements unterstützt

Planungseinheit	auch Planningunit; Gruppen für die Kap.- und Schulungsplanung des Cockpits; definiert durch Flugzeugtyp, Funktion, Homebase, Fluggesellschaft, Unterfunktion
Unterfunktion	auch Subfunction; optionale Erweiterung der Funktion

Anlagenverzeichnis