Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Hendrik Behrens

Integration einer RFID-basierten Vorflugkontrolle für Notfallausrüstung in ein Kabinenmanagementsystem

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Hendrik Behrens

Integration einer RFID-basierten Vorflugkontrolle für Notfallausrüstung in ein Kabinenmanagementsystem

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Fohl
Zweitgutachter: Prof. Dr. God

Eingereicht am: 17. Juni 2016

Hendrik Behrens

Thema der Arbeit
Integration einer RFID-basierten Vorflugkontrolle für Notfallausrüstung in ein Kabinenmanagementsystem

Stichworte
RFID, Vorflugkontrolle, CIDS, CCS, FAP, Inventarisierung, Notfallausrüstung, Flugzeugkabine

Kurzzusammenfassung
Diese Arbeit beschreibt die Erstellung eines RFID-basierten Systems zur Erfassung von Notfallausrüstung in der Flugzeugkabine. Es werden RFID Tags angebracht an der Notfallausrüstung mit einen Mobilen RFID Lesegerät erfasst und mit Hilfe von WLan an das CIDS übertragen. Auf dem FAP werden die eingescannten Daten auf einer neu erstellen Seite ausgegeben und können in einem Log abgelegt werden.
Hendrik Behrens

Title of the paper
Integration of an RFID-enabled Preflight-Check for Emergency Equipment into a Cabin Management System

Keywords
RFID, Pre-Flight-Check, CIDS, CCS, FAP, Inventory, Emergencyequipment, Aircraftcabin

Abstract
This work describes the creation of an RFID-based system for the detection of emergency equipment inside the aircraft cabin. RFID tags attached to the emergency equipment is read with a mobile RFID reader and transferred using WiFi at the CIDS. At the FAP the scanned data is output on a new page and can be stored in a log.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Abkürzungsverzeichnis

A4A Air for Amerika

ANSI American National Standards Institute

API Application Programming Interface

ATA Air Transport Association

CCS Cabin Core System

CIDS Cabin Intercommunication Data System

DIN Deutsches Institut für Normung

DCL Digial Cabon Logbook

EPC Electronic Product Code

FAP Flight Attendant Panel

FKS Institut für Flugzeug-Kabinensysteme

GS1 Global Standards One

ISO International Organization for Standardization

IEC International Electrotechnical Commission

RFID Radio-Frequenz-Identifikation

MTP Media Transfer Protocol

SAE Society of Automotive Engineers

SysML Systems Modeling Language

Tag Transponder

UHF Ultra High Frequenzy

UML Unified Modeling Language

WLAN Wireless Local Area Network

WSDL Web Services Description Language

USB Universal Serial Bus

ZAL Hamburgs Zentrum für Angewandte Luftfahrtforschung

1. Einleitung

Die Logistikbranche nutzt bei ihren Prozessen heute zunehmend die Radio-Frequenz-Identifikation (RFID)-Technik zum automatischen und berührungslosen Identifizieren von Waren und Gütern. Dazu werden RFID-Transponder (Tag) auf dem jeweiligen Objekt angebracht, mit denen per Funk kommuniziert werden kann. Eine direkte Sichtverbindung zum Lesegerät ist nicht notwendig und es können eine Vielzahl an Tags quasi zeitgleich erfasst werden. Auch ermöglicht diese Technologie das Speichern zusätzlicher Informationen am Objekt, so dass neben einer individuellen Kennung, d.h. einer ID, z.B. zusätzlich das Herstell- und das Haltbarkeitsdatum gespeichert und übertragen werden können.

Auch in der Flugzeugbranche setzen die Flugzeughersteller und Ausrüster mittlerweile auf eine Markierung von Bauteilen und Ausrüstungsgegenständen mit RFID-Tags. Hierbei geht es einerseits um die Verbesserung der Prozessdokumentation bei der Produktion und der späteren Wartung und Instandhaltung. Andererseits wird durch den Einsatz von RFID auch beim täglichen Betrieb eine Optimierung von Arbeitsaufgaben möglich. So müssen vor jedem Start eines Flugzeugs aufwändige Preflight-Checks durchgeführt werden. In der Flugzeug-Kabine ist dazu unter anderem die Kontrolle der dort befindlichen Notfallausrüstung von den Luftfahrtbehörden verpflichtend vorgeschrieben. Die Verfügbarkeit und Benutzbarkeit der Ausrüstung wird gegenüber der sogenannten Minimum Equipment List geprüft, vergleichbar mit der Prüfung des Inhalts eines Verbandskastens im Auto. Diese Sichtkontrolle nimmt reichlich Zeit in Anspruch und die Kontrolle der Schwimmwesten unter den Sitzen bereitet einige Mühe, weil die zu kontrollierenden Objekte nicht alle einfach zugänglich sind. Innerhalb festgelegter Wartungszyklen müssen neben der Verfügbarkeit zusätzlich auch die Haltbarkeitsdaten für die Ausrüstung kontrolliert werden.

Es gibt daher bereits erste Ansätze, Wartungsaufgaben und Preflight-Checks durch den Einsatz von RFID-Technik zu vereinfachen, jedoch stellen aktuell verfügbare Systeme, beispielsweise von den Firmen Lufthansa Technik oder EAM, Standalone-Lösungen dar, die nicht in das zentrale Kommunikationssystem des Flugzeugs integriert sind. Aufgrund dieser fehlenden

Anbindung an das Kabinenmanagementsystem, Cabin Core System (CCS), kann das Optimierungspotential durch RFID sowohl bei der Anzeige auf dem Flight Attendant Panel (FAP), der zentralen Benutzerschnittstelle des CCS, als auch beim Prozessablauf selbst nicht vollständig genutzt werden.

1.1. Aufgabenstellung

Vor diesem Hintergrund wird in dieser Bachelorarbeit ein System entworfen und als Demonstrator aufgebaut, das eine Optimierung der Preflight-Checks für die Notfallausrüstung durch Nutzung von RFID-Technologie ermöglicht. Hierzu müssen zunächst die bestehenden Prozesse und die Systemumgebung analysiert werden, um daraus die Anforderungen an das zu entwerfende System abzuleiten. Werkzeugtechnisch und methodisch ermöglicht eine modellbasierte Erfassung der Anforderungen sowie der Entwurf der Systemarchitektur mit Hilfe von formalen Sprachen wie der Systems Modeling Language (SysML) oder der Unified Modeling Language (UML) ein strukturiertes Vorgehen.

Beim zu entwerfenden System soll ein tragbares RFID-Lesegerät zum Einsatz kommen, mit dem die Kabinenbesatzung beim Durchgang durch die Kabine die gesamte Notfallausrüstung über die daran angebrachten RFID-Tags erfasst. Die Ergebnisse dieses Preflight-Checks sollen anschließend an das CCS übermittelt werden und dort weiter verarbeitet und ausgewertet werden. Kann ein Erfassungsvorgang ohne Beanstandung abgeschlossen werden, sollen für die Notfallausrüstung automatisiert ein partieller „Cabin-Ready“-Status gemeldet und eine dazugehörige Dokumentation für das Logbuch erstellt werden. Fehler und Warnungen sollen im CCS bearbeitet und zur weiteren Klärung bei Bedarf wieder an das tragbare RFID-Lesegerät übertragen werden können.

Das Lesegerät benötigt für diesen Vorgang eine interne Datenbank für die Zwischenspeicherung der Daten, eine einfache grafische Bedienoberfläche und die Anbindung an ein Funknetzwerk (Wireless Local Area Network (WLAN)) zum Datenaustausch. Auf einer primären Datenbank im CCS sollen Informationen über die Notfallausrüstung gespeichert werden, welche gegen die Leseergebnisse abgeglichen werden müssen.

Das Lesegerät benötigt für diesen Vorgang eine interne Datenbank für die Zwischenspeicherung der Daten, eine einfache grafische Bedienoberfläche und die Anbindung an ein Funknetzwerk (WLAN) zum Datenaustausch. Auf einer primären Datenbank im CCS sollen Infor-

mationen über die Notfallausrüstung gespeichert werden, welche gegen die Leseergebnisse abgeglichen werden müssen.

Als tragbares RFID-Lesegerät soll das Produkt „Morphic“ des Herstellers Nordic ID verwendet werden. Das CCS wird durch einen so genannten Airbus-A350-CIDS-Demonstrator repräsentiert. Ausführliche Systemtests am fertigen Demonstrator sollen das Zusammenspiel aller Komponenten zeigen und eine Validierung der Anforderungen ermöglichen.

2. Stand der Wissenschaft und Technik

Diese Arbeit setzt sich mit vielen Technologien und Abläufen auseinander, die bereits Verbreitung in der Industrie und Wissenschaft gefunden haben. Dieses Kapitel gibt eine Einleitung zu Entwicklung und Stand dieser Technologien. Eingangs betrachtet werden jene Abläufe der Vorflugkontrolle, die in dieser Arbeit optimiert werden sollen. Außerdem gibt es einen Überblick über Entwicklung, grundlegende Funktion und heutige Einsatzgebiete, besonders im Flugzeug, von Radio-Frequenz-Identifikation, welche in der Arbeit eingesetzt werden soll. Die für RFID eingesetzte Hard- und Software, besonders das gewählte Nordic ID Lesegerät, wird ebenfalls betrachtet. Im RFID Umfeld ist es außerdem notwendig sich, mit den vielen Standards und Normen auseinanderzusetzen, die diese Technologie definieren.

Eine Komponente auf Flugzeugseite, die weitere Betrachtung benötigt, ist das CCS oder bei Airbus Cabin Intercommunication Data System (CIDS) genannt. Dazu gehört das FAP, an dem Erweiterungen vorgenommen werden sollen. Das zur Planung und Modellierung eingesetzte UML wird ebenfalls aufgegriffen. Hierzu gehört die Entwicklungsumgebung Cameo Systems Modeler.

2.1. Vorflugkontrolle der Notfallausrüstung

Bei jedem Start eines Flugzeuges müssen umfangreiche Checklisten erfüllt werden, um einen sicheren Flug zu gewährleisten. Diese Checkliste umfasst alle Bereiche eines Flugzeuges, somit auch den Bereich der Passagierkabine. Die in dieser Arbeit relevante Liste nennt sich „Minimum Equimpent List“ und beschreibt was im Flugzeug mitgeführt werden muss, um starten zu dürfen. Die sogenannte „Master Minimum Equipment List“[MME16] wird von den Behörden in Zusammenarbeit mit dem Flugzeughersteller erstellt und umfasst alles das im Flugzeug mitgeführt werden muss, um in dem Einzugsbereich dieser Behörden fliegen zu dürfen. Airlines erweitern diese Liste meistens um eigene Ausrüstung und erstellen damit ihre eigene „Minimum Equipment List“[RJM12].

Diese Arbeit soll einen Teil dieser Kontrolle, den der Notfallausrüstung in der Flugzeugkabine, vereinfachen. Zu dieser Notfallausrüstung gehören unter anderem ausreichend Rettungswesten, Erste-Hilfe Koffer und eine Notaxt. Diese müssen alle vorhanden und funktionstüchtig sein, damit das Flugzeug starten darf. Das Vorhandensein muss vor jedem Start kontrolliert werden. Bei Rettungswesten und Erste-Hilfe Koffer müssen zusätzlich Haltbarkeitsdaten beachtet werden. Hierbei gibt es Unterscheidungen zwischen Ausrüstungen, die vor dem Start verpflichtend vorhanden sein müssen und welche, die auch bei Gelegenheit ersetzt werden können. Beispielsweise werden einige Rettungswesten mehr mitgeführt als es Personen im Flugzeug gibt, um bei Beschädigungen Ersatz zu haben.

Für eine Kontrolle ist diese Ausrüstung nicht immer leicht erreichbar platziert und kann besonders bei großen Flugzeugen mit vielen Sitzplätzen sehr viel Zeit in Anspruch nehmen. Diese Zeit die nicht genutzt werden kann um Passagiere zu transportieren kann sich schnell zu einen großen Kostenfaktor entwickeln, so dass jede Beschleunigung dieses Prozesses bares Geld wert ist.

Für diese Aufgabe ist RFID bestens geeignet, denn Optimierung von Inventarisierungsprozessen ist bei dieser der größte Einsatzbereich. Zusätzlich von Vorteil ist, dass viele Objekte in der Luftfahrt bereits in Produktion oder der Lieferkette mit RFID Tags ausgestattet werden, die weiterhin verwendbar sind. Bisherige Studien haben eine Arbeitszeitreduzierung von bis zu 90 % gezeigt[STU10].

Abbildung 2.1.: Preflight-Checks mittels RFID-markierter Ausrüstung [RFI13]

2.2. Die RFID Technologie

Die Abkürzung RFID steht für den englischen Begriff „radio-frequency identification“ und bedeutet soviel wie „Identifikation durch Funkwellen“. Dies ist eine Technik, die es ermöglicht, Objekte mit kleinen Chips und eingebauter Antenne, die Transponder oder auch RFID-Labels genannt, auszustatten, mit denen diese dann aus der Entfernung zu erkennen sind. Hierbei wird von einem Lesegerät ein Signal ausgesendet, auf das Tags mit einem eigenen Signal antworten. Der Vorteil durch dieses System ist die Erkennung ohne Sichtkontakt und über Distanzen von mehreren Metern. Die Erkennung benötigt nur wenige Millisekunden und in näherer Umgebung kann das Lesegerät auch hunderte Tags gleichzeitig verarbeiten. So können innerhalb weniger Sekunden so viele Objekte erkannt werden, das ohne Hilfe von RFID Minuten oder sogar Stunden dauern würde. Dies lässt sich teilweise komplett automatisieren, sodas zum Beispiel alle Waren die ein Lagerhaus verlassen erkannt werden ohne das auch nur ein Mensch etwas dafür tun muss.

Unterschieden wird hier zwischen passiven und aktiven Tags. Passive Tags besitzen keine eigene Stromversorgung und für eine Antwort genügt das Signal des Lesegerätes. Diese Art von Tags ist am weitesten verbreitet und sehr günstig herzustellen. Aktive Tags verfügen hingegen über eine eingebaute Energiequelle. Durch diese zusätzliche Energie ist es möglich diese Tags über deutlich größere Reichweiten auszulesen als passive Tags.

Diese deutlich komplexere Bauweise macht aktive Tags allerdings auch um ein vielfaches teurer und deutlich größer. Während passive Tags oft nur wenige Cent kosten, sind dies bei aktiven Tags mehrere Euro. Diese passiven Tags haben üblicherweise eine Dicke von unter einen Millimeter, wo hingegen für aktive Tags mehr als fünf Millimeter nicht ungewöhnlich sind. Als Beispiel, während die passiven Tags an Produkten selbst angebracht zu tausenden in einen Container transportiert werden, ist der Container selbst mit einen aktiven Tag ausgestattet um diesen auch über größere Entfernung erkennen zu können.

Außerdem ist es möglich auf Tags Daten zu speichern. Während einfache Tags nur ihren Electronic Product Code (EPC), eine mindestens 64 Bit, üblicherweise aber 96 oder 204 Bit, lange Identifikationsnummer speichern, können manche Tags mehrere Kilobyte an Daten aufnehmen. Die gespeicherten Daten können von entsprechenden Lesegeräten auch verändert werden. Beispielsweise während eines Transportes werden alle Stationen, eingeschlossen

Bearbeiter und Datum, gespeichert, um später den Weg zurückverfolgen zu können falls dieses notwendig ist.

2.2.1. Entwicklung von RFID

Die Geschichte der RFID Technik geht gemessen an anderen Funktechnologien sehr weit in die Vergangenheit. Der erste Vorläufer dieser Technologie wurde schon im Zweiten Weltkrieg von dem schottischen Physiker Robert Alexander Watson-Watt entwickelt, um britische Kampfflugzeuge in der Erkennung untereinander zu helfen.Diese ersten Systeme waren noch sehr groß und unhandlich, kaum zu vergleichen mit der heutigen Technik. Koffer große Geräte in den Flugzeugen strahlten aktiv Signale aus, um von Verbündeten erkannt zu werden.

Erst nach dem Krieg, im Jahre 1948, wurde die Technik zur passiven Erkennung wie wir sie heute kennen erstmals beschrieben. Ab den 60er Jahren fand die RFID Technologie erstmals als Diebstahlschutzsystem den Weg in die zivile Nutzung. Einfache RFID Sender wurden an Artikeln angebracht und beim Verlassen der Läden von Lesegeräten an den Ausgängen erkannt, was einen Alarm auslöst. In den 70er Jahren folgten Tierkennzeichnungssysteme in der Landwirtschaft und ein erstes System, das Schlüssel an Türen ersetzt.[RFJ15][GB06][RFT07]

Die 90er brachten Ultra High Frequenzy (UHF) Tags, welche Reichweiten von mehr als 6 Metern bei passiven Systemen ermöglichte. Zu einer weiten Verbreitung von Identifikationskarten, beispielsweise in Skipässen, Tankkarten oder Zugangskontrollen kam es zu dieser Zeit auch erstmals. Ebenfalls erwähnenswert sind die in diesen Jahren aufkommenden RFID basierte Mautsysteme.

Doch erst nach der Jahrtausendwende gab es einen wirklich großen Preisverfall durch den der großflächigen Einsatz in viel größeren Stückzahlen, auch an günstigen Artikeln, erlaubten. Nun war die Verbreitung so weit fortgeschritten, dass erste Normen notwendig wurden für einen einfacheren universalen Einsatz. Die wichtigsten waren der EPC, der 2003 fertiggestellt wurde und für eine weltweit eindeutige Zuteilung von RFID Tags ermöglichte. Die verwendeten Funkfrequenzen wurden 2004 in der ISO 18000[ISO15] festgelegt.

Der Markt für RFID wächst seit Jahren mit großer Geschwindigkeit. Waren es 2005 noch 565 Mio. verkaufte Tags[CIO06], waren es 2006 schon ca. 1,3 mrd[CWR06]. Auch der Umsatz der Branche stieg von 3,7mrd im Jahre 2007[si206] auf geschätze 10,4 mrd 2015[GO115].

In der heutigen Zeit nimmt die Verwendung von RFID noch immer rasant zu und immer neue Anwendungsbereiche werden gefunden. Bargeldloses bezahlen oder digitale Visitenkarten werden heute über immer ausgefeiltere RFID Technik möglich.

2.2.2. Heutiger Einsatz von RFID

Heute hat RFID viele Anwendungsbereiche gefunden und wird im großen Maßstab in vielen Bereichen eingesetzt wie der beachtliche Branchenumsatz zeigt[GO115], welcher noch weiter stark anwächst. Derzeitig gibt es einige besonders große Einsatzbereiche.

- Logistik
- Identifikation
- Bezahlsysteme

Einsatz von RFID in der Logistik ist heutzutage der Haupteinsatzbereich. Produkte werden direkt in der Herstellung mit RFID Tags versehen um Transport und Verkauf zu vereinfachen. Behälter müssen während des Transports nicht mehr geöffnet werden um den Inhalt zu kontrollieren, was Arbeit und Zeit spart. Noch heute gibt es hier viel Potenzial für Verbesserung, da es trotz vieler Normen noch wenige einheitliche Kennzeichnungen in der gesamten Transportkette gibt. Bei diesem Einsatz ist der Preis eines einzelnen Tags besonders wichtig. Wenn es um viele, günstige Artikel geht, können schon wenige Cent für ein Tag den Preisvorteil zunichtemachen.

Identifikation kann es für Menschen, Tiere und auch Objekte geben. Für Maut Systeme werden alle Fahrzeuge mit einen RFID Tag ausgestattet und können so automatisch beim Passieren von Kontrollstationen erfasst werden. Bei Tieren werden die RFID Chips oft unter der Haut eingesetzt, wodurch diese nicht mehr durch Scheuern verloren gehen können. Die kontaktlose Arbeitsweise von RFID ermöglicht hier den Einsatz. Bei Menschen bedeutet Identifikation meist Zutrittskontrolle. Die RFID Tags kommen hier oft in Form von Karten die an Türen oder anderen Eingängen eingelesen werden müssen, um Zutritt zu erhalten. Bei diesem Einsatz ist der Vorteil von RFID, dass für jede Person individualisierte Zugänge vergeben werden können. Durch solche Zugänge ist es möglich genauere Zugangsregeln festzulegen, als es mit herkömmlichen Schlüsseln möglich wäre. Bei Verlust des RFID Chips ist ein Sperren und Ersetzen ohne großen Aufwand möglich. Abschließend ist es mit RFID auch möglich zu ermitteln, wer sich wann in bestimmten Bereichen aufgehalten hat.

2.2.3. Einsatz von RFID im Flugzeug

Bereits heute wird RFID in der Luftfahrt in vielen Bereichen eingesetzt. Schon im Flughafen beginnt es damit das Gepäckbeförderung durch Tags im Barcodeetikett erweitert werden.

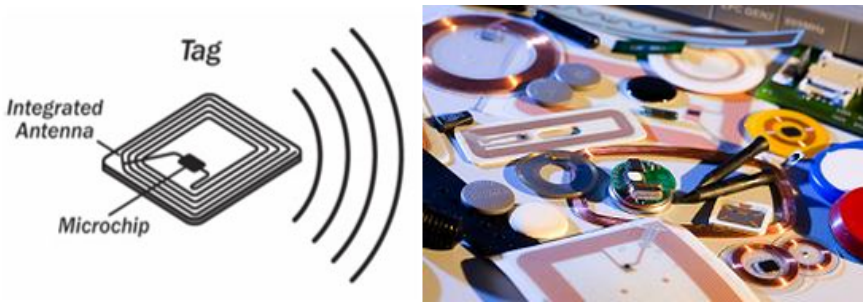

Abbildung 2.2.: Symbolischer Aufbau eines RFID Tags und einige verbreitete Bauformen [RFI16] [RFI14]

Lesefehler der Sortiermaschinen wie diese oft bei Barcodes vorkommen und zu Fehlleitungen des Gepäcks führen werden so stark reduziert. Im Flugzeug selbst wird RFID genutzt um besonders Wartungsvorgänge zu erleichtern und zu beschleunigen. Seit 2009 stattet Airbus alle Austauschbaren, Reparablen und in der Lebensdauer begrenzte Flugzeugteile des A350 XWB und seit 2013 Sitze und Rettungswesten aller Modelle mit Tags aus.[RFA12] Da ein Flugzeug für RFID und Funktechnik allgemein ein besonderer Bereich in dem es viele bestehende Normen und Richtlinien gibt, müssen Tags gewählt werden die diese einhalten. Es gibt Vorschriften für die Funkwellen diese ausstrahlen, bis hin dazu wie Daten auf Tags abgelegt werden.

2.2.4. RFID Standards

Die RFID-Technologie hat heute viele Standards, die für einen reibungslosen Einsatz beachtet werden müssen. Die folgenden Normen und Standards sind die wichtigsten die diese Arbeit betreffen.

ISO-IEC 18000-6.2013

Die International Organization for Standardization (ISO) und die International Electrotechnical Commission (IEC) bilden ein gemeinsames technisches Komitee. Nationale Institute wie das Deutsches Institut für Normung (DIN) oder American National Standards Institute (ANSI) aus den USA sind Mitglieder und arbeiten dort zusammen in Gremien und Arbeitsgruppen an der Entwicklung international einheitlicher Standards.

Die ISO-IEC 18000-6.2013[ISO15] ist eine Norm für die Spezifikation von innerhalb von Frequenzbändern für RFID. Der 6. Teil dieser Norm beschreibt Kommunikation mit passiven Tags auf Frequenzen von 860-930 MHz, welche in dieser Arbeit eingesetzt werden. Ein Frequenzbereich wird verwendet um in unterschiedlichen Ländern zu ermöglichen die Frequenz

so zu wählen, dass es keine Störungen durch Frequenzüberlappungen gibt. Tags die diese Norm einhalten sind in der Lage International eingesetzt zu werden ohne das es zu Konflikten kommt. Die genauen Funktionen beschrieben durch die ISO-IEC 18000-6.2013 Norm sind:

- Identifizierung und Kommunikation mit mehreren Transpondern im Feld,
- Auswahl einer Untergruppe von Transpondern zur Identifizierung oder mit dem sie kommunizieren,
- Lesen, Schreiben oder mehrmaliges Wiederbeschreiben von Daten auf ein einzelnen Transponder,
- Benutzerkontrollierte dauerhaft verschlüsselbare Speicher,
- Datenintegritätsschutz,
- Kommunikationsverbindung vom Lesegerät zum Transponder mit Fehlererkennung,
- Kommunikationsverbindung vom Transponder zum Lesegerät mit Fehlererkennung,
- Unterstützung für passive Backscatter-Transponder mit oder ohne Batterie.

GS1 EPCglobal Class 1 Gen 2 Version 1.2.0

Global Standards One (GS1)[EPC08] ist eine internationale, privatwirtschaftliche Organisation für die Schaffung von Standards von Produkten zur eindeutigen Kennzeichnung von Anlagen, Behältern, Dokumenten und anderen Business Objekten. EPCglobal ist der auf RFID-Technik fokussierte Firmenteil, dem über 700 Unternehmen angehören und auf die Entwicklung und Standardisierung von RFID-Technologie ausgelegt ist. Class 1 spezifiziert:

- Physikalische Vorgänge,
- Abläufe und
- Befehle
- zwischen Lesegeräten und passiven Tags im Frequenzbereich von 860-960MHz.
- Kollisionshandling in Umgebungen mit mehreren Tags.

ISO 18000-6 entspricht weitgehendst dem EPCglobal Class 1 Gen 2, da dieser auf diesem aufbauend entwickelt wurde. Tags sind daher in der Lage beide Standards einzuhalten.

ATA Spec2000.2009 Chapter 9

Der Spec2000[Spe09] enthält Techniken und Standards, die weltweit vom Großteil der Luftfahrtindustrie akzeptiert werden um so eine optimale Grundlage für Kommunikation und Datenaustausch bereitzustellen. Es handelt sich hierbei nur um eine Empfehlung, da diese allerdings von vielen Behörden und Organisationen als Grundlage für Normen und Richtlinien verwendet wird, ist die Einhaltung zu empfehlen.

Erstellt wurde der Spec2000 von der Air for Amerika (A4A), früher unter dem Namen Air Transport Association (ATA), einem Dachverband amerikanischer Fluggesellschaften.

Das Kapitel 9-5 der Spec2000 betrifft Anforderungen für RFID Tags die zur dauerhaften Kennzeichnung von Teilen in Flugzeugen eingesetzt werden. Diese Tags sollen Teilerückverfolgbarkeit des gesamten Produktlebenszyklus gewährleisten. Die Struktur des Speichers auf dem Tag wird im Anhang 11 beschrieben.

Wichtige Inhalte der Spec2000 Kapitel 9-5 sind:

- Dauerhafte Anbringung von Tags an Flugzeugteilen.
- Anwendung für Wartung und innerbetriebliche Logistik.
- Übergreifenden Einsatz, vom Teilezulieferer über Flugzeughersteller bis zu den Fluggesellschaften.
- Flexible und Zukunftssicherer Aufbau.
- Verwendung des ISO-IEC 18000-6.2013

SAE AS5678-2006

Society of Automotive Engineers (SAE) ist eine gemeinnützige Organisation für den Fortschritt der Mobilitätstechnologie.

Die SAE AS5678-2006[SAE06] ist eine Anforderungsspezifikation für passive RFID Tags welche in der Luft- und Raumfahrtindustrie eingesetzt werden sollen.

Zentrale Anforderungen sind:

- Anforderungen für passive UHF RFID Tags nutzbar in der Luft- und Raumfahrtindustrie.
- Minimale Anforderungen für eine Nutzung von Tags am Boden
- Grund liegende Anforderungen für eine Nutzung im Flug die weiter in der RTCA DO-160E beschrieben werden.
- Anwendung existierender Standards.
- Festlegung eines Standards der dauerhafte Anbringung an Flugzeugteilen erlaubt.

2.2.5. Das Nordic ID RFID Lesegerät

In dieser Arbeit wurde ein RFID Lesegerät des Herstellers Nordic ID verwendet. Eine Beschreibung der Eigenschaften dieses Lesegerätes und wie diese genutzt werden.

Zum lesen von RFID Tags nach ISO18000-63 wird ein geeignetes Lesegerät benötigt, welches selbst geschriebene Programme ausführen kann. Zur Verbindung mit dem Kabinensystem muss eine drahtlose Verbindung möglich sein, Display und Eingabemöglichkeiten zur Benutzung eines kleinen Programms muss möglich sein.

Im Institut wurde hierfür bereits das RFID Lesegerät der Serie Morphic von dem Hersteller Nordic-ID angeschafft, das alle Anforderungen dieser Arbeit erfüllt und sich daher für den Einsatz in dieser eignet.

Das Nordic-ID Morphic ist ein kleines, handliches Lesegerät im Taschenformat welches über einen kleinen Touchscreen und WLAN verfügt. Es läuft basierend auf einen Microsoft Windows Embedded CE 6.0 Betriebssystem und kann auf integrierten Flashspeicher oder einsteckbarer microSD Karte Software aufnehmen. Schnittstellen zur Nutzung der Geräte Funktionen sind die Nur**Api!** und die MHL-Schnittstelle.

Abbildung 2.3.: Nordic ID Morphic RFID Handterminal [nor16]

Im späteren Verlauf der Arbeit ist das Model Nordic-ID Medea hinzugekommen. Bei dem Medea Model handelt es sich um ein Handheld RFID Lesegerät das im Vergleich zum Morphic Model über ein deutlich größeres Display verfügt. Zusätzlich zu höherer Auflösung bring dieses Modell deutlich bessere Bedienbarkeit nur mit Fingern. Bei dem kleineren Morphic Modell

ist es bei vielen Bedienelementen des Windows CE notwendig einen Stift zu benutzen. Das Inventarisierungsprogramm aus dieser Arbeit soll möglichst ohne Stift bedienbar sein.

Abbildung 2.4.: Nordic ID Medea UHF RFID Cross Dipole [nor16]

Nordic ID nurAPI

Bei der Nutzung des Nordic ID RFID Lesegerätes wird die „NurAPI“ genannte Application Programming Interface (API) genutzt. Eine Beschreibung der wichtigsten Funktionen

Ein wichtiges Element bei der Programmierung von Programmen für RFID Lesegeräte ist der Zugriff auf Hardwareelemente wie dem RFID Empfänger. Um diesen Zugriff bei den Nordic-ID Lesegeräten zu ermöglichen, hat der Hersteller hierfür eine Schnittstelle mit dem Namen nurAPI entwickelt die auf allen derzeitigen und zukünftigen Lesegeräten eine einheitliche Nutzung ermöglichen. Programme die auf einem Gerät programmiert wurden lassen sich dadurch auch auf anderen Geräten ohne Anpassung verwenden. Nur Unterschiede in der Hardwareausstattung müssen bedacht werden. Beispielsweise sind nicht alle RFID Lesegeräte auch mit einen Barcode Scanner oder mit WLAN ausgestattet. Diese API Funktionen würden auf diesen Geräten keine Funktion haben.

2.3. Cabin Core System

Abbildung 2.5.: FAP und CIDS Demonstrator des FKS

Ein wichtiger Teil dieser Arbeit ist die Integration des RFID Erfassungssystems in bereits im Flugzeug vorhandene Systeme. Dies soll die Nutzbarkeit verbessern und weitere Funktionen ermöglichen. Bei diesem System handelt es sich um das Cabin Core System, das zentrale Steuersystem der Passagierkabine im Flugzeug. Hier werden unter anderem Funktionen wie Licht, Lüftung und Durchsagen der Mannschaft gesteuert. Die Schnittstelle zu der Mannschaft stellt das Flight Attendant Panel da.

2.3.1. Was sind CIDS und FAP

Bei den zu erweiternden Komponenten handelt es sich um das FAP des CIDS. CIDS ist die Bezeichnung, welche bei Airbus für das CCS genutzt wird. Der Demonstrationsaufbau des Institut für Flugzeug-Kabinensysteme ist auf Abbildung 2.5 zu sehen. Der Schrank links im Bild repräsentiert einen Flugzeugausschnitt in dem das FAP, sichtbar als Touchscreen-monitor, eingelassen ist. Bei dem weitverbreiteten A320 Modell beispielsweise können Passagiere diesen Monitor beim Boarding hinter den vorderen Türen sehen[BOA15]. Das FAP Modell des Institut für Flugzeug-Kabinensysteme (FKS) basiert auf dem im A350 eingesetzten System. Bei dem

FAP handelt es sich um einen eigenständigen Rechner mit integrierten Touchscreen, welcher eine Netzwerkverbindung mit dem CIDS hat.

Auf der rechten Seite von Abbildung 2.5 ist zu sehen ein vollständiger Demonstrationsaufbau des A380 CIDS. Dieser umfasst den Director (linker Rack, mittlere Höhe), ein FAP, Sprechanlagen, Beleuchtung, Medienserver und Schalter für Bedienelemente an den Passagiersitzen.

Bei dem in dieser Arbeit verwendeten FAP handelt es sich um einen Demonstrator, der in der Programmiersprache Java [JAV16] programmiert wurde. Dieser Demonstrator kopiert Design und Funktionalität möglichst originalgetreu, läuft allerdings ohne spezielle Hardware und kann standalone, ohne Kommunikation zu anderen Elementen, arbeiten.

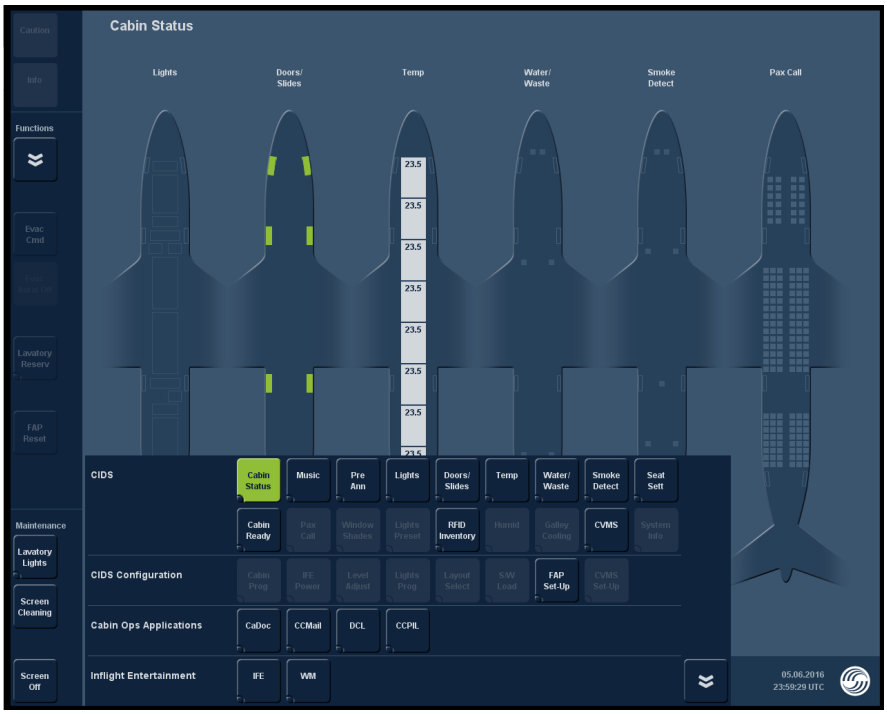

Abbildung 2.6.: Bedienoberfläche des FAP Demonstrator

2.3.2. PDLs für das FAP Design

Die grafische Oberfläche des FAP Demonstrators wird in einem proprietären erstellt. Der gesamte Aufbau der Bedienoberfläche wird in der fap.pdl Datei definiert. In dieser werden Sequenziell alle Elemente aufgebaut. Es werden Seiten Definiert, diesen IDs und Elemente hinzugefügt. Jedes Element erhält ebenfalls IDs, Position, Größe und gegebenenfalls Beschriftungen. Diese pdl kann gegebenenfalls von Hand angepasst werden oder mit Hilfe von m4

Makros generiert werden. Eine Ausnahme bildet hier die Menüleiste zum Aufruf der verschiedenen Funktionsseiten. Diese ist vollständig in Java programmiert und muss auch dort verändert werden.

Das Design des FAP ist in einer umfangreichen Styleguide von Airbus vorgegeben die von dem FAP Demonstrator genaustens umgesetzt wird. Weitere wichtige Dateien in der die Oberfläche definiert wird sind die fapen.txt, welche alle Texte und Beschriftungen in englischer Lokalisierung enthält und die fap.res Datei in welcher Pfade zu Bild- und Tondateien und Farbwerte gespeichert sind. In der Abbildung **??** ist das FAP Design in der Hauptübersichtsseite mit ausgeklappten Menü zu sehen. Die Standartauflösung des FAP ist 1280x1024 Pixel.

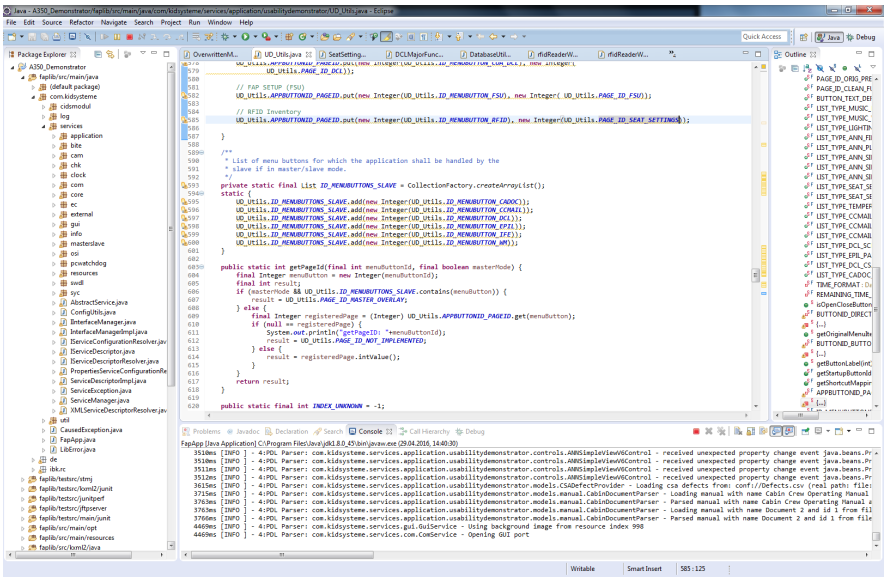

Abbildung 2.7.: Eclipse Entwicklungsumgebung mit geöffneten FAP Project

2.3.3. Director-Client Konzept des FAP

Ein wichtiger Aspekt bei der Programmierung des FAP Demonstrators ist das Director-Client-Konzept, welches zum Sicherheitskonzept gehört. Das im Flugzeug eingesetzte FAP besitzt keine eigene Logik, sondern gibt alle erhaltenen Steuerbefehle an den Director weiter welcher den Kern des CIDS darstellt. Dieser Director gibt daraufhin Befehle zurück, wie sich die Oberfläche des FAP zu verändern hat. Im FAP Demonstrator wurde dieses Konzept ebenfalls umgesetzt und ist protokolltechnisch in der Lage mit einem Director gekoppelt zu werden. Der Demonstrator enthält allerdings einen eigenen Director, welcher standalone Betrieb ermöglicht,

was Entwicklung und Testen deutlich vereinfacht. Dieses Konzept ist notwendig um die hohen Sicherheitsstandards in der Flugfahrt einzuhalten. Durch die Auslagerung aller Logik in den abgeschotteten und redundant ausgelegten Director werden wichtige Systeme des Flugzeuges vor Angriffe von außen geschützt und Ausfallsicherheit erhöht. Einen Überblick über den Systemaufbau gibt Abbildung 2.8

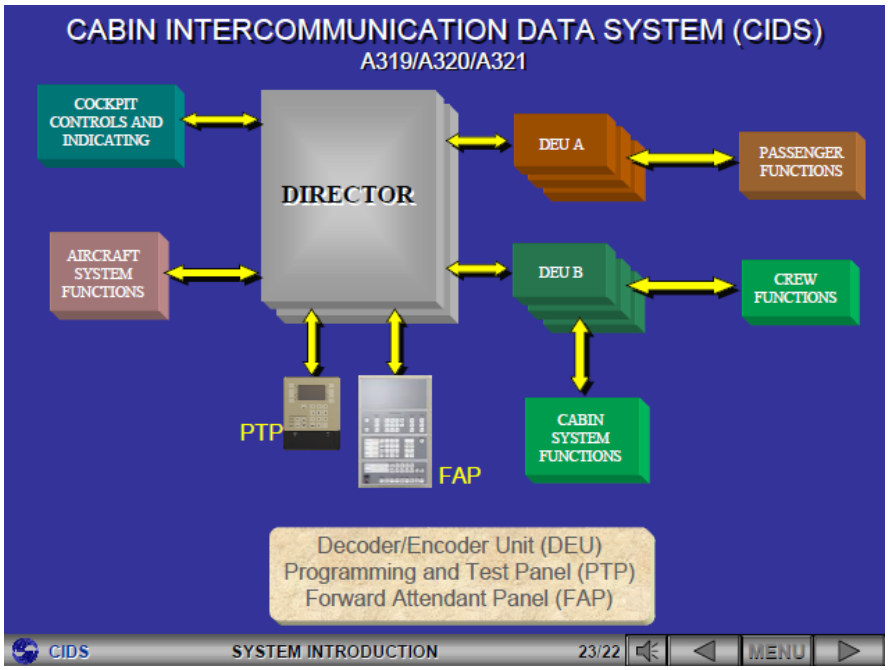

Abbildung 2.8.: Aufbau Schematik des CIDS [CID16]

2.4. UML

Um große Projekte geordnet und effektiv bearbeiten zu können, ist es notwendig diese mit einer ausgiebigen Planung zu beginnen. In dieser Planung müssen alle Funktionen die das fertige Produkt besitzen soll festgelegt werden und darauf aufbauend wie dieses erreicht werden soll.

Um dies zu tun, gibt es sogenannte Modellierungssprachen. Im FKS wird für diesen Zweck UML und dessen Dialekt SysML verwendet. Als Modellierungsumgebung findet das Programm Cameo System Modeler 2.9 Anwendung. Cameo ist eine Java basierte und hoch flexible Umgebung, die angefangen bei der Erstellung von UML Diagrammen wie Klassen-, Use-Case- oder Aktivitätendiagrammen hilft, als auch komplexe Simulationen der späteren Abläufe ermöglicht. Eingeschränkt ist es auch möglich direkt Code aus den Diagrammen auszuführen und so bereits existierende Programme aus Modellen heraus steuern zu können.

Die in dieser Arbeit zu Modellierenden Teile sind Use-Case-Diagramme für die notwendigen Anwendungsfälle wie das Scannen der Notfallausrüstung oder das erstellen eines Logs. Außerdem Klassendiagramme zum logischen Aufbau der Programme und ein Modell der Datenbanken.

Abbildung 2.9.: Cameo Systems Modeler Symbolbild

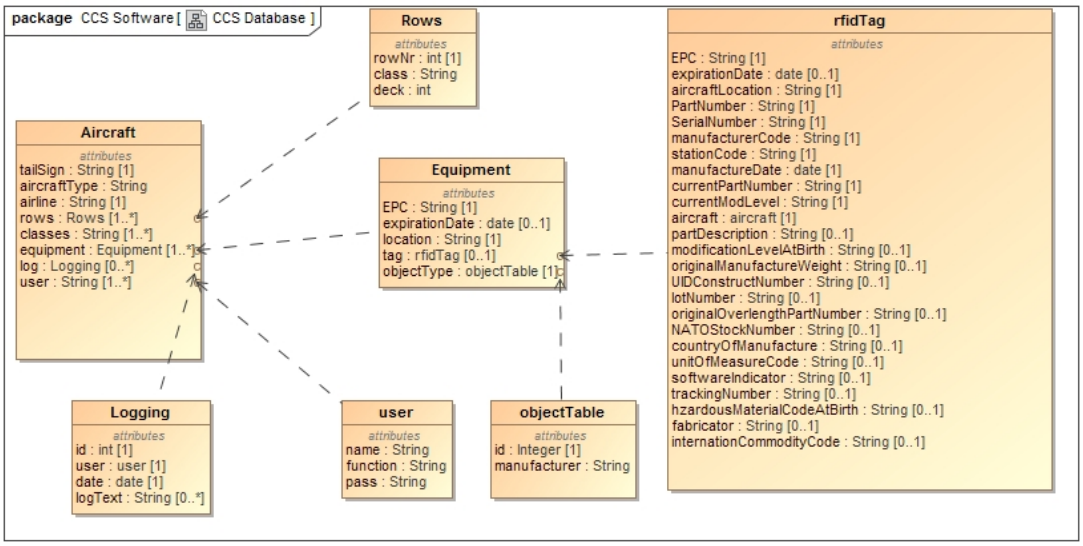

Abbildung 2.10.: Modell der Datenbank zur Speicherung der Flugzeug, Equipment und Tag Daten

3. Ausarbeitung

Nach Festlegung der Ziele dieser Arbeit und ausgiebiger Analyse der gegebenen Ausgangssituation kann mit der Ausarbeitung begonnen werden. Es muss eine Softwarearchitektur erstellt werden, daraufhin die Softwareprojekte selbst entwickelt und abschließend auf die vorher aufgestellten Anforderungen getestet werden.

3.1. Planung

Der zweite Schritt in der Entwicklung eines Softwareprojektes ist der Entwurf der Software Architektur. Hierfür müssen Nutzungsabläufe geplant werden und für die Umsetzung dieser Programmelemente vordefiniert werden. Ebenfalls müssen mögliche Implementierungsmöglichkeiten gefunden und gegeneinander abgewogen werden.

3.1.1. Abläufe

Um die nötigen Programmfunktionen zusammenstellen zu können, ist es anfangs notwendig alle Arbeitsschritte die mit dem fertig Programm ausgeführt werden sollen nachzuvollziehen. In diesem Fall wäre dies ein vollständiger Inventarvorgang zur Vorflugkontrolle. Dieser Ablauf gliedert sich in etwa folgenderweise auf:

- Den Nordic ID Morphic Handscanner für den Vorgang vorbereiten. Einschalten, Programm starten, Inventory Modus wählen.
- Den Inventory Vorgang starten und die Notfallausrüstung in der Kabine einscannen.
- Nach dem Einscannen der gesamten Notfallausrüstung alle Daten über WLAN an das FAP übertragen.
- Auf dem FAP alle aufgenommenen Daten auf Korrektheit kontrollieren.
- Den Vorgang abschließen indem ein Eintrag im Log erstellt wird.

Bei diesem Ablauf kann es natürlich zu Fehlern kommen, die entsprechend behandelt werden müssen. Mögliche Fehlerfälle wären nicht eingescannte Ausrüstung, beispielsweise durch schlechten Funkkontakt oder durch das Fehlen von Ausrüstung. In diesem Fall muss die Crew darüber informiert werden und die Möglichkeit haben entweder erneut zu scannen oder fehlende Ausrüstung zu ersetzen. Ebenfalls kann es zu Verbindungsfehlern zwischen RFID Lesegerät und dem FAP durch schlechte WLAN Verbindung kommen. In diesem Fall muss es möglich sein die Daten erneut zu übertragen. Eine weitere Nutzung, die bedacht werden muss, ist das Ändern von gespeicherter Ausrüstung durch die Wartungsmannschaft. Beispielsweise das ersetzen von abgelaufenen Schwimmwesten durch neue.

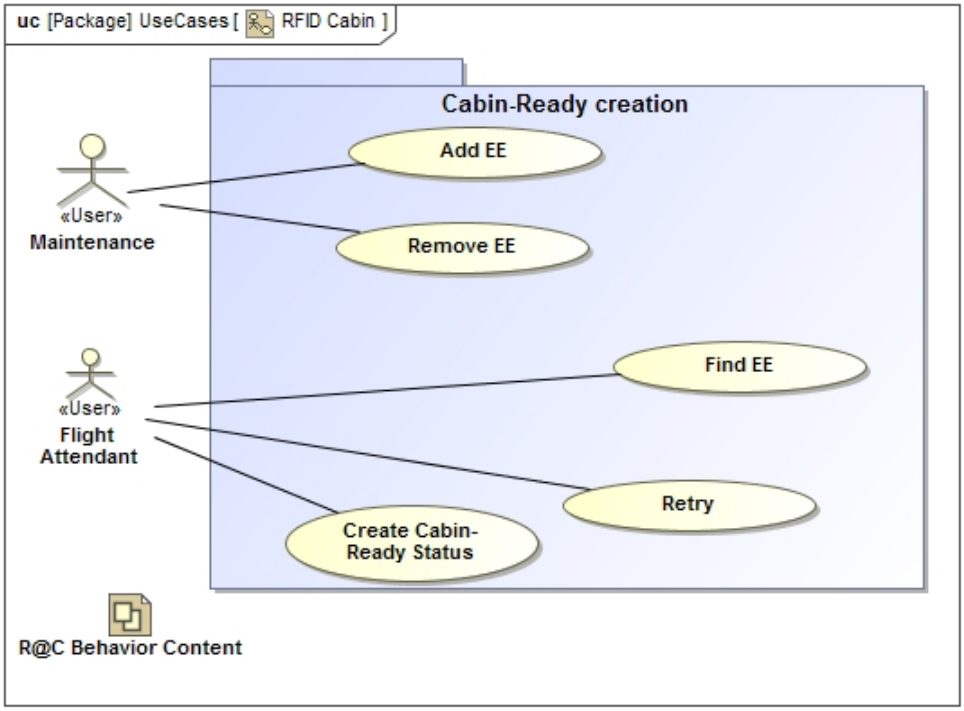

Abbildung 3.1.: Use-case-Diagram möglicher Aktionen in diesem Projekt

3.1.2. Programmaufbau/Planung

Die zu programmierenden Teile dieser Arbeit lassen sich in drei Bereiche einteilen. Dem RFID Lesegerät mit Oberfläche, Scannerlogik und der Kommunikation zum FAP. Das FAP selbst mit Oberfläche, Netzwerkserver, Steuerlogik und Erstellung von Log Einträgen. Die Datenbanken in Lesegerät und FAP sollten zur einfacheren Programmierung und Wartung

möglichst identisch sein und daher für beide zusammen erstellt werden. Diese Teile werden in unterschiedlichen Programmiersprachen und Hardware bearbeitet und sollten daher schon in der Planung getrennt betrachtet werden und über gemeinsame Interfaces verbunden werden.

Architektur des RFID Lesegerätes

Während das Windows Embedded CE 6.0 des Nordic ID Morphic die Unterstützung für viele verschiedene Programmiersprachen bietet, wird diese durch die Schnittstellen und Softwareunterstützung von Nordic ID auf die Programmiersprache „C#“ limitiert.

Bei der Wahl der Kommunikationsart über das WLAN fällt die Entscheidung auf die Web Services Description Language (WSDL), da es in diesem Bereich bereits Vorkenntnisse und als Beispiel nutzbare Programme für diese Anwendung gibt. WSDL ist eine Plattform unabhängige Beschreibungssprache zum Austausch von Daten auf XML Basis. Um einen Datenaustausch über WSDL zu erreichen, benötigt es ein vordefiniertes Interface. Für eine Verbindung zwischen RFID Lesegerät und FAP welche die gewünschten Funktionen erfüllt, muss das Interface folgende Aktionen ausführen können:

- Eine Ping Funktion zum testen der Verbindung.
- Das Übertragen der EPCs von gescannten Tags.
- Eine Funktion um einen erneuten Scan vom Lesegerät anzufordern.
- Die Möglichkeit eine Liste der erneut zu scannenden Tags auf das Lesegerät schicken zu können.

Das WSDL Interface definiert nach diesen Anforderungen ist in Abbildung 3.2 zu finden.

Abbildung 3.2.: Interface für die WSDL basierte Kommunikation zwischen Lesegerät und FAP

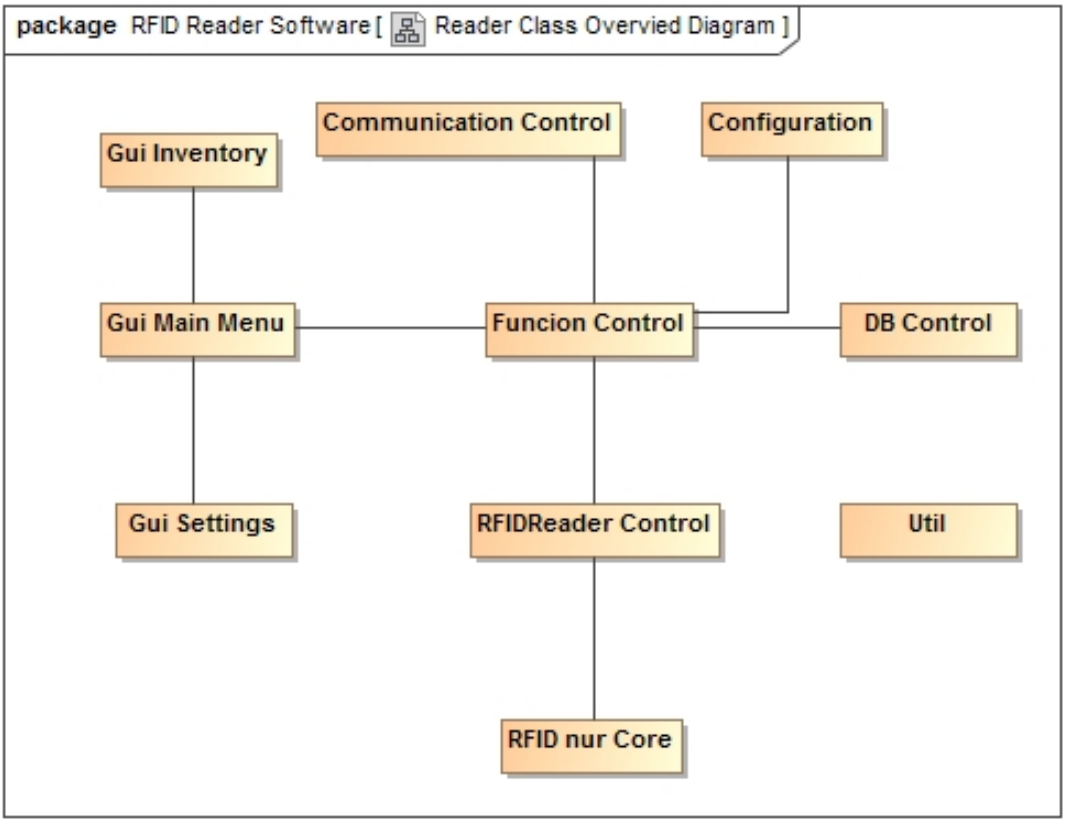

Abbildung 3.3.: Class Diagram des RFID Lesegerätes

Architektur des FAP Demonstrators

Die Architektur des FAP Demonstrators ist weitgehend durch den bereits existierenden Code gegeben. Hier ist der Schwerpunkt dabei diese Architektur zu analysieren, um die neuen Seiten und Funktionen an den korrekten Stellen vorzunehmen. In der Abbildung 3.4 lässt sich eine vereinfachte Übersicht über deren Aufbau finden. Das FAP ist intern streng in mehrere Bereiche getrennt, welche in separaten Threads ausgeführt werden und über eine interne Schnittstelle Datenpakete austauscht. Das Format dieser Pakte ist ein proprietärer Standard von Airbus, dessen Aufbau in dieser Arbeit nicht analysiert werden muss. Ein Hauptthread ist dafür zuständig beim Systemstart Resourcedateien wie die PDL auszulesen und daraus die Oberfläche zu generieren. Im laufenden Betrieb hat dieser Thread zwei Aufgaben. Bei Eingaben auf dem Touchscreen des FAP das Objekt an der gewählten Stelle zu bestimmen und dessen ID auf der internen Schnittstelle zu senden. Außerdem nimmt dieser Thread Nachrichten über

Änderungen der Oberfläche entgegen und passt seinen Status entsprechend an.

Im anderen Hauptthread des FAP Demonstrators befindet sich die Logik, welche die Aktionen ausführt. Dieser Thread simuliert die Funktion des Directors und kann über Konfigurationseinstellungen vollständig abgeschaltet werden. In diesem Fall muss die interne Schnittstelle aus dem Demonstratorprogramm herausgeführt werden und mit einem weiteren Demonstrator oder einem Director verbunden werden.

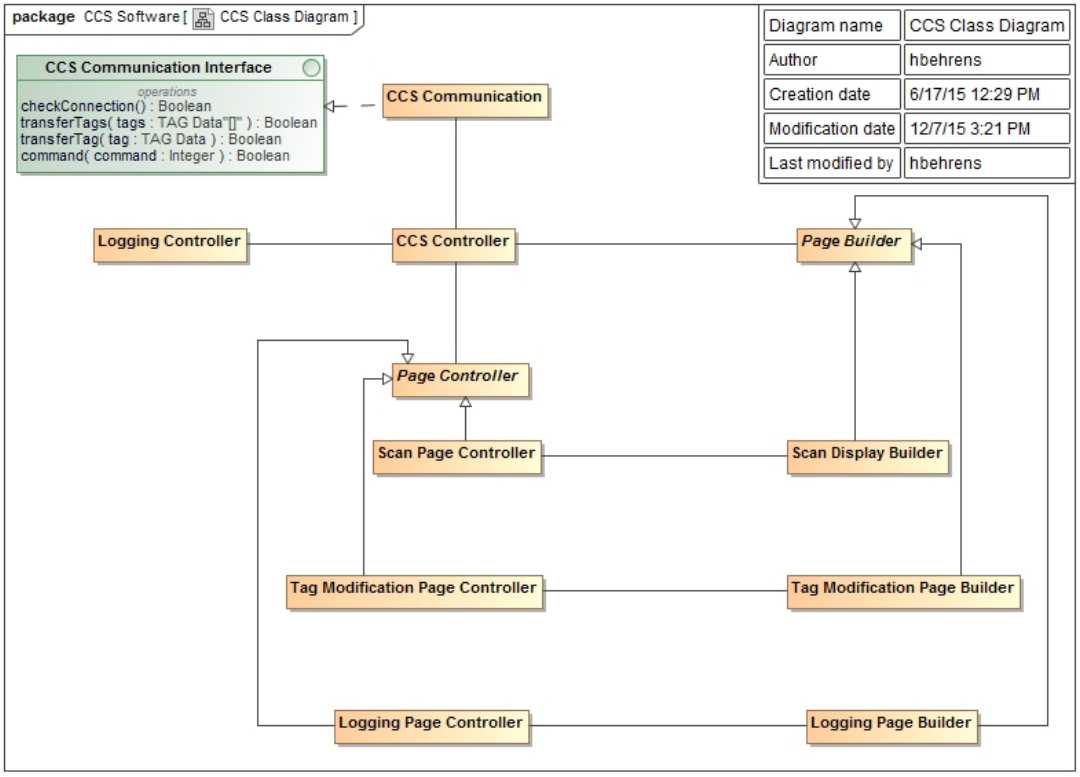

Abbildung 3.4.: Stark vereinfachtes Class Diagram des FAP Demonstators zur Verdeutlichung der internen Abläufe

3.2. RFID Lesegerät

In dieser Arbeit wurde ein RFID Lesegerät des Herstellers Nordic ID verwendet. Diese Entscheidung ist gefallen, da dieses Gerät bereits im Institut vorhanden war und es aus vorhergegangenen Projekten Vorwissen gibt. Dieses Gerät basiert auf dem Microsoft Windows

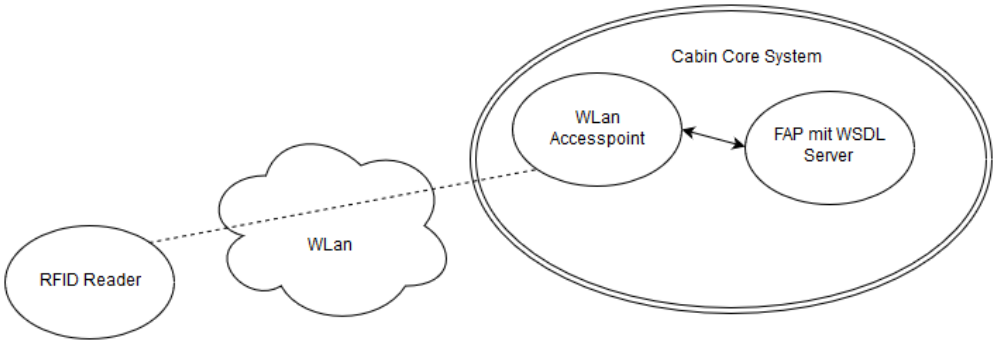

Abbildung 3.5.: Symbolischer Aufbau der Netzwerkverbindung zwischen RFID Lesegerät und FAP

Embedded CE 6.0 Betriebssystem und setzt zur Erstellung von Programmen Microsoft Visual Studio der Versionen 2005 oder 2008 voraus. Neuere Versionen von Visual Studio bieten keine Unterstützung für das Windows Embedded CE 6.0. Daher fällt die Entscheidung über die Entwicklungsumgebung für den RFID Lesegerät auf Microsoft Visual Studio 2008, der aktuellsten nutzbaren Version.

Einrichten der Entwicklungsumgebung für das RFID Lesegerät

Bei dem Einrichten von Visual Studio zur Programmierung des RFID Lesegerätes gibt es einige Schritte zu beachten. Nach der Installation von Visual Studio 2008 entsprechend der Anweisungen des Microsoft Installers muss ein spezieller Treiber von Nordic ID installiert werden. Dieser Treiber wird bei dem Lesegerät mitgeliefert und hat den Namen „Morphic_SDK_2011.msi“'. Bei der Installation ist zu beachten, dass es einen Fehler in Verbindung mit aktuellen Windows Versionen gibt, durch den die Installation mit Standardeinstellungen abbricht. Um diesen Fehler zu umgehen muss die „Custom“ Installation ausgewählt und darin die Dokumentation abgewählt werden.

Das Nordic ID Morphic Lesegerät muss mit Hilfe der dazugehörigen Ladestation und einem USB Kabel mit dem Rechner verbunden werden. Bei der erstmaligen Verbindung installiert Windows üblicherweise selbstständig das „Windows Mobile Gerätecenter 6.1“. Solle dies nicht geschehen, muss dies manuell nachgeholt werden. Sobald das Morphic Lesegerät korrekt installiert wurde, bestehen zwei parallele Verbindungen über die USB Verbindung. Einmal werden die Datenspeicher des Lesegerätes über das Media Transfer Protocol (MTP) im System eingebunden, wodurch direkter Zugriff auf die Festspeicherbereiche des Lesegerätes möglich

wird. Außerdem wird eine virtuelle TCP/IP Verbindung über USB eingerichtet, was Remote Steuerung und das Testen von Netzwerkfunktionen ohne WLAN-Verbindung möglich wird. Die IP-Adresse des Morphic Lesegerätes ist standardmäßig: 192.168.100.1. Eine wichtige Eigenschaft der Nordic ID Lesegeräte ist, dass das Betriebssystem in einem flüchtigen Speicher arbeitet und bei einem vollständigen Neustart zurückgesetzt wird. Dateien die erhalten bleiben sollen müssen im Flashspeicher des Lesegerätes abgelegt werden. Zur dauerhaften Speicherung von Betriebssystemeinstellungen wie WLAN-Verbindung, Idlezeiten für Stromsparfunktionen oder Programme im Autostart muss nach Einrichtung dieser ein Systemabbild erstellt werden und dieses so eingerichtet werden, dass es bei Systemstart geladen wird. Damit ist die Einrichtung der RFID Lesehardware abgeschlossen.

Beim Anlegen des Entwicklungsprojektes in Visual Studio ist es wichtig von beginn an die richtigen Einstellungen zu wählen. Spätere Änderung ist nicht, oder nur unter großen Aufwand möglich. Für das Nordic ID Morphic Lesegerät ist die korrekte Einstellung ein Visual C# Projekt für Smart Devices. Nach dem Einrichten muss eventuell noch das korrekte Zielgerät in der Menüleiste von Visual Studio ausgewählt werden. Das korrekte Gerät trägt die Bezeichnung: „Morphic SDK ARMV4I Device“. Wenn nun das Projekt in Visual Studio ausgeführt wird und das RFID Lesegerät mit dem Rechner verbunden ist, startet das sich in Entwicklung befindende Programm automatisch auf dem Lesegerät. Um dieses ohne Rechneranbindung starten zu können, müssen die kompilierten Programmdateien manuell auf den Flashspeicher des Lesegerätes kopiert werden und dort ausgeführt werden.

3.2.1. Funktionen des Lesegerätes/API

Die grundlegende Funktion des RFID Lesegerätes, das Auslesen von Tags, wird über das ansprechen der NurAPI von Nordic ID erfüllt. Diese API ermöglicht den Zugriff auf tiefgreifende Hardwarefunktionen, benötigt eine Initialisierung bei Programmstart und tiefgehende Kenntnisse über Lese- und Schreibvorgänge mit RFID. Zum vereinfachten Zugriff auf die Funktionen der NurAPI gibt es eine auf diese aufbauende Klasse, im Projekt „RFID nurCore“ genannt. Von der „RFID nurCore“ Klasse wird zum Programmstart eine Instanz erstellt, die statisch jederzeit vom gesamten Programm nutzbar ist. Beim Beenden vom Programm muss darauf geachtet werden, dass die Instanz korrekt disposed wird, damit alle genutzten Ressourcen wieder freigegeben werden. Die NurAPI ermöglicht es nicht das mehrere Programme zeitlich auf die Hardware des Lesegerätes zugreifen. Die „RFID nurCore“ Klasse ist mit dem Ziel entwickelt später auch in anderen Anwendungen auf dem RFID Lesegerät nutzbar zu sein und die Erstellung eines RFID Tag scannenden Programms mit geringen Aufwand ermöglichen.

Dies wurde im FKS bereits getan und hat die Entwicklung eines einfachen Tag scannenden Programms innerhalb eines Arbeitstages ermöglicht. Die Austauschbarkeit zwischen verschiedenen Lesegeräten des Herstellern Nordic ID welche die NurAPI unterstützen ist mit diesem Projekt ebenfalls erfolgreich getestet. Der Aufbau dieser Klassen ist in dem Klassendiagramm der Abbildung 3.6 zu sehen.

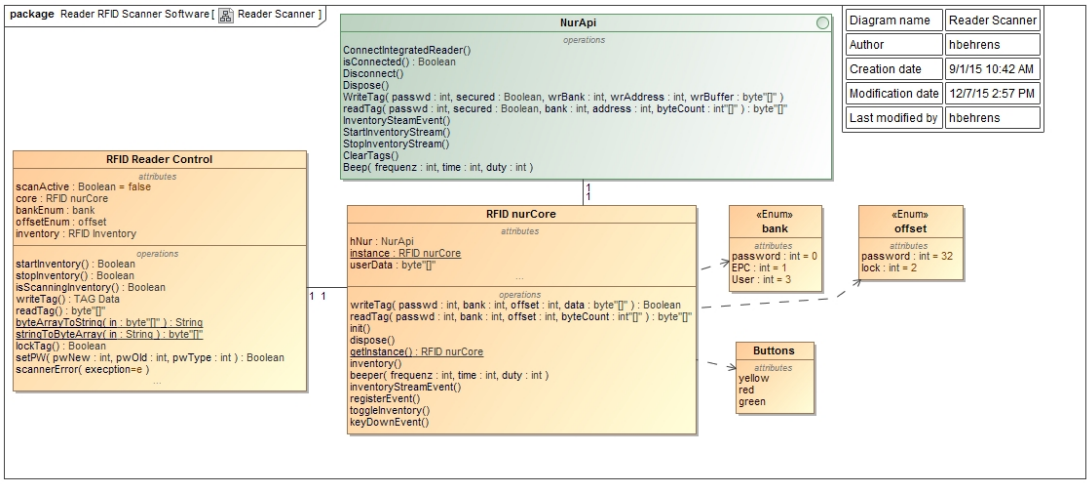

Abbildung 3.6.: Klassendiagramm das des den RFID Lesegerätes steuernden Programms (Größere Version im Anhang)

Die für dieses Projekt wichtigste Funktion ist die integrierte Inventarisierungsfunktion. Zur Nutzung dieser Funktion muss das Programm auf den Eventbus der „RFID nurCore“ Klasse registriert werden. Dieses Event löst während eines laufenden Inventarvorganges bei jedem gefundenen Tag aus und übermittelt jeweils einen EPC. Die Registrierung auf dem Eventbus und Start/Stopp des Inventarvorganges geschieht mit folgendem Code:

```
1 private RFIDreaderControl reader = new RFIDreaderControl();
2 reader.getCore().CoreEvent +=
3 new RFIDNurCore.TickHandler(this.inventory_Event);
4 reader.toggleInventory(true);
5 //Hier läuft der Inventarvorgang
6 reader.toggleInventory(false);
7 reader.getCore().Clear();
8 //Inventarvorgang abgeschlossen und bereits gescannte Tags gelöscht.
```

3.2.2. GUI

Die Oberfläche des Programms des RFID Lesegerät wird in Visual Studio mit Hilfe des Windows Form-Designer erstellt. Das Design soll der FAP Styleguide folgen und ohne Stift nur mit den Fingern zu bedienen sein. Es gibt starke Einschränkungen durch den nur 320x240 Pixel großen Bildschirm. Das alternative Lesegerät des FKS, der Nordic ID Medea besitzt ein größeres Display mit 800x480 Pixeln, steht aber während der Hauptentwicklungszeit nicht zur Verfügung. Aufgrund der identischen Software und API Plattform ist allerdings davon auszugehen das ein späteres Portieren mit geringen Aufwand möglich ist.

Bei dem Erstellen der Seiten hat sich gezeigt, dass es im .Net Framework 3.5, welches unter Visual Studio zum Programmieren des Lesegerätes genutzt wird starke Einschränkungen gibt. Die Funktion Buttons Bilddateien zuzuweisen ist hier nicht vorhanden. Daher muss von der vorhandenen Button-Klasse eine ImageButton-Klasse abgeleitet werden. In dieser abgeleiteten Klasse muss die OnPaint Methode überschrieben werden. Dies wird folgenderweise erreicht [IMG03]:

```
1  public class ImageButton : Control
2  {
3      private Image image;
4      private bool bPushed;//flag to indicate the pressed state
5      private Bitmap m_bmpOffscreen;
6   protected override void OnPaint(System.Windows.Forms.PaintEventArgs
       e)
7   {
8      Graphics gxOff;   //Offscreen graphics
9      Rectangle imgRect; //image rectangle
10     Brush backBrush;   //brush for filling a backcolor
11     if (m_bmpOffscreen == null) //Bitmap for doublebuffering
12     {
13        m_bmpOffscreen = new Bitmap(ClientSize.Width, ClientSize.
             Height);
14     }
15     gxOff = Graphics.FromImage(m_bmpOffscreen);
16     gxOff.Clear(this.BackColor);
17     if (!bPushed)
18         backBrush = new SolidBrush(Parent.BackColor);
19     else //change the background when it's pressed
20         backBrush = new SolidBrush(Color.DarkBlue);
```

```
21      gxOff.FillRectangle(backBrush, this.ClientRectangle);
22      if (image != null)
23      {
24          //Center the image relativelly to the control
25          int imageLeft = (this.Width - image.Width) / 2;
26          int imageTop = (this.Height - image.Height) / 2;
27          if (!bPushed)
28          {
29              imgRect = new Rectangle(imageLeft, imageTop,
30              image.Width, image.Height);
31          }
32          else //The button was pressed
33          {
34              //Shift the image by one pixel
35              imgRect = new Rectangle(imageLeft + 1, imageTop + 1,
36              image.Width, image.Height);
37           }
38          //Set transparent key
39          ImageAttributes imageAttr = new ImageAttributes();
40          imageAttr.SetColorKey(BackgroundImageColor(image),
41          BackgroundImageColor(image));
42          //Draw image
43          gxOff.DrawImage(image, imgRect, 0, 0, image.Width, image.
                Height,
44          GraphicsUnit.Pixel, imageAttr);
45
46          using (Font myFont = new Font("Arial", 14,FontStyle.Regular)
                )
47          {
48              StringFormat sf = new StringFormat();
49              sf.Alignment = StringAlignment.Center;
50              sf.LineAlignment = StringAlignment.Center;
51              gxOff.DrawString(this.Text, myFont,
52              new SolidBrush(Color.White), imgRect, sf);
53          }}
54          if (bPushed) //The button was pressed
55          {//Prepare rectangle
56              Rectangle rc = this.ClientRectangle;
57              rc.Width--;
58              rc.Height--;
```

```
59              //Draw rectangle
60              gxOff.DrawRectangle(new Pen(Color.Black), rc);
61          }
62          //Draw from the memory bitmap
63          e.Graphics.DrawImage(m_bmpOffscreen, 0, 0);
64          base.OnPaint(e);
65      }
66  }
```

In Abbildung 3.7 zu sehen, ist das Hauptmenu für den RFID Lesegerätes. Farbwahl, Schriftart und Buttons entsprechen exakt der Airbus Styleguide für das FAP. Es gibt außerdem Vorgaben für Abstände und Positionen, diese sind bei dem begrenzten Platz des Handgerätes allerdings nicht umsetzbar.

Die Optionen sind für eine einfache Bedienbarkeit auf das Nötigste reduziert. Das Hauptmenü bietet nur die Möglichkeit des Öffnens des Inventorymenüs, eines Einstellungsmenü für Sendestärken von RFID und WLAN Antenne und dem „Exit“ Button zu korrekten Beenden des Programms. Sollte das Programm auf anderen Weg beendet werden, kann es unter Umstanden dazu führen, dass nicht alle Hardwareressourcen freigegeben werden und erst nach einen Neustart des RFID Lesegerätes wieder genutzt werden können.

Das Inventorymenü(Siehe Abbildung 3.8) zeigt nach dem Öffnen den gespeicherten Sitzplan und Buttons um den Scanvorgang zu starten oder zum Hauptmenü zurückzukehren. Sollte das Flugzeug mehr Sitzplätze beziehungsweise mehr Notfallausrüstung haben als auf der verfügbaren Bildschirmfläche angezeigt werden kann, werden dieser scrollingfähig. Alle Ausrüstungsgegenstände können durch Drücken der ihnen zugeteilten Buttons alle über diese gespeicherten Informationen anzeigen. Diese Anzeige schließt sich durch ein erneutes Drücken wieder. Auf dem Nordic ID Morphic Gerät ist das Auswählen der Ausrüstungsgegenstände aufgrund des kleines Bildschirms ohne Stift sehr umständlich. Bei Nutzung des Nordic ID Medea Gerätes zeigt sich hier ein deutlicher Vorteil der größeren Bedienfläche.

Die programmiertechnische Umsetzung der Sitzübersicht ist eine List von „ACRow“ Objekten beziehungsweise deren Ableitungen für unterschiedliche Sitzkonfigurationen. In der „ACRow“ Klasse wird die Verwaltung von bis zu acht Notfallausrüstungsgegenständen übernommen. Mehr als 8 Sitze in einer Reihe kommen zwar in Großflugzeugen vor, lassen sich aber nicht auf dem Bildschirm des RFID Lesegerätes unterbringen. Außer der Anzeige wird in dieser Klasse auf das Aufrufen der genaueren Ausrüstungsinformationen reagiert und das

Abbildung 3.7.: Displaydarstellung des Inventarisierungsprogramms auf dem RFID Lesegerät im Hauptmenü nach dem Start

Einfärben entsprechend des Status vorgenommen. Für die verschiedenen Konfigurationen der Reihen wird die „ACRow“ Klasse abgeleitet. Die derzeitigen Konfigurationen sind Economy, Business und Box, für den Arztkoffer, bei welchen nicht genutzte Buttons ausgeblendet werden und die Verbleibenden auf passende Positionen verschoben werden.

Der Scanvorgang mit dem Lesegerät gestaltet sich simpel. Nachdem „Start“ gedrückt wurde, wird jedes RFID Tag in Lesereichweite erfasst und dessen EPC mit der Datenbank abgeglichen. Wird eine Übereinstimmung gefunden, wird die dazugehörige Notfallausrüstung im Sitzplan aktiviert und über das WLan eine Benachrichtigung an das FAP gesendet. Der Lesevorgang wird mit einen Druck auf „Send“ beendet, wodurch alle bisher gefundenen Tags und ein Schlusssignal erneut übertragen werden.

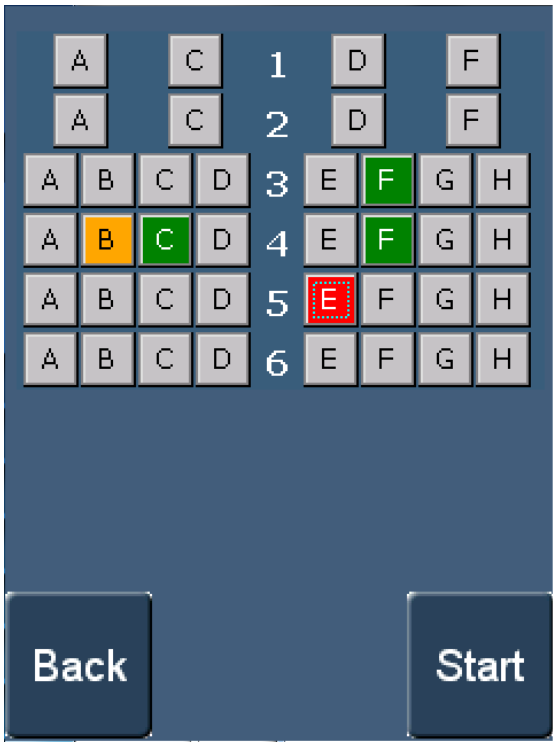

Abbildung 3.8.: Displaydarstellung während des Inventarisierungsvorganges mit ersten erfassten Tags

3.2.3. Netzwerkkommunikation

Das Einrichten einer WLAN Verbindung mit dem CIDS ist unabhängig von dem Inventarisierungsprogramm in der Windowsoberfläche vorzunehmen.

Um unter Microsoft Visual Studio eine WSDL Verbindung einzurichten, muss dem Projekt eine „Web Reference“ hinzugefügt werden. Hierfür muss die Adresse des WSDL Servers mit den gewünschten Verbindungsinformationen angegeben werden. Visual Studio erstellt aus den bereitgestellten Verbindungsdaten selbstständig eine API und alle für die Verbindung notwendigen Klassen. In der Generierten „Reference.cs“ finden sich die notwendigen Methoden mit denen es möglich ist, auf den Server zuzugreifen und gegebenenfalls die Zieladresse anzupassen, sollte sich diese in anderen Systemen unterscheiden.

3.3. Datenbanken

Die in diesem Projekt genutzte Datenbankversion ist „Microsoft SQL Server Compact 3.5“, da diese sowohl unter Windows Embedded CE 6.0 als auch unter Java nutzbar ist. Es wird in beiden Programmen, auf dem Lesegerät als auch im FAP die selbe Datenbank verwendet, da größtenteils die gleichen Daten gespeichert werden müssen. Die Planung sieht es vor das auf der Datenbank des FAP auch die Logfiles der Scans abgelegt werden. Dies entfällt, da es mit dem Digial Cabon Logbook (DCL) bereits einen integrierten Speicherort für Logs gibt.

3.3.1. Tabellenaufbau

Der Tabellenaufbau ist entsprechend der Abbildung 2.10.
Es ist der Sitzaufbau des Flugzeuges gespeichert und die Zuweisungen von Ausrüstung an dessen Lagerorte. Zu Ausrüstung sind alle Informationen gespeichert, die zur Funktion des Programms notwendig sind. EPC, Ablaufdatum, Standort im Flugzeug und um welchen Typ von Ausrüstung es sich handelt. Zusätzlich gibt es eine Tabelle für Logeinträge, diese wird allerdings nicht genutzt. Die Tabelle für zusätzliche Tag Informationen orientiert sind an der ATA Spec2000.2009 Chapter 9[Spe09], wo diese für Tags in der Luftfahrt gedacht sind.

3.3.2. Nutzung der Datenbanken

Beim Programmstart auf dem RFID Lesegerät werden die Informationen über Sitzreihen und Ausrüstung aus der Datenbank ausgelesen und darauf basierend die Flugzeugübersicht generiert. Im FAP geschieht dies nicht, da dort die Oberfläche mit PDLs generiert wird und sich nicht flexibel ändern lässt.

In beiden Programmen wird die Datenbank genutzt, um bei Aufruf eines Ausrüstungsgegenstandes, weitere gespeicherte Informationen auszugeben. Ebenfalls wird beim Einlesen eines neuen Tags in der Datenbank nach dessen EPC gesucht und sollte dieser gefunden werden liefert die Datenbank die Informationen zu welchen Platz diese gehört.

3.4. FAP

Der FAP Demonstrator ist ein Programm entwickelt für Airbus, um für den neuen A350 Tests und Optimierungen in möglichst originalgetreuer Funktion durchführen zu können, bevor es eine Flugfähige Version von diesem gab. Der Demonstrator wird mit Java in einer Eclipse Entwicklungsumgebung programmiert. Die vorgenommenen Erweiterungen erfordern

Integration in bestehende Strukturen und teilweise das erweitern oder ändern vorhandener Klassen. Folgend gibt es einen Überblick über diese Änderungen.

Einrichten der Entwicklungsumgebung für den FAP Demonstrator

Um am FAP Demonstrator arbeiten zu können, ist es notwendig die Entwicklungsumgebung Eclipse zu installieren[ecl16]. Das Entwicklungsprojekt liegt als gepackter Eclipse Workspace vor und kann nach dem Entpacken geöffnet werden. Für die Nutzung ist es notwendig einige weitere Bibliotheken einzubinden. Dabei handelt es sich um die Folgenden:

- commons-codec-1.3.jar
- commons-logging.jar
- forms-1.0.7.jar
- jmf.jar
- RXTXcomm.jar
- xerces-2.4.0.jar
- xml-apis.jar
- xmlrpc-2.0.jar

Zur Generierung der PDL Dateien ist es zusätzlich notwendig Cygwin zu installieren, um notwendige Linuxtools unter Windows nutzen zu können. Um den FAP Demonstrator außerhalb der Eclipse Entwicklungsumgebung nutzen zu können, muss aus den vorliegenden Class files und resourcen eine nutzbare jar Datei erstellt werden. Dies geschieht mit dem Eclipse Plugin Ant[ant16] für das ein Buildscript im Projekt vorliegt.

3.4.1. Einfügen neuer Bedienseiten

Das Einfügen neuer Seiten in den FAP Demonstrator beginnt mit dem Generieren neuer PDL Dateien. Die dafür notwendigen Sourcedateien und Scripte finden sich unter:
../Workspace/A350_Demonstrator /SW_Development/fap/implementation/GUILayout/sources

Im Unterordner */AppPages* finden sich alle m4 Dateien zur Generierung der einzelnen Seiten des FAP Demonstrator. Die neue Seite ist eine Kopie von „SeatSettingsClassic.m4“ Bei der alle nicht benötigten Buttons, Sitze und Beschriftungen entfernt sind. Die ursprünglichen Callreset

Buttons sind ersetzt durch „Save Log“, „Rescan Missing“ und „Edit Inventory“ ersetzt. Unter *../GUILayout/res* sind in der Datei „fapen.txt“ die neuen Bezeichnungen hinzugefügt. Da alle IDs intern mit Lücken vergeben wurden, ist es ausreichend jeweils um eins hoch zuzählen, um noch nicht vergebende zu finden. Der Code der neu erstellten Seite befindet sich im Anhang.

Zur Generierung werden nacheinander die „mkpdl.bat“ und die „build_pdl.cmd“ ausgeführt. Die neu generierten Dateien werden automatisch in das Eclipse Buildverzeichnis verschoben, sodass die Änderungen nach dem nächsten Neustart sichtbar sind.

Die einzige Benutzeroberflächenänderung, die nicht durch die PDL beeinflusst wird ist das Menü. Inaktive Buttons des Menus können in der Klasse „com.kidsysteme.services.application.usabilitydemonstrator.UD_Utils.java“ aktiviert und mit einer Seite verknüpft werden. Änderungen sind:

```
1 Line 165: public static final int PAGE_ID_RFID_INVENTORY = 0x36;
2 Line 335: UD_Utils.BUTTONID_BUTTONTEXT.put(new Integer(0x130d01ff),
     "RFID\nInventory");
3 Line 589: UD_Utils.APPBUTTONID_PAGEID.put(new Integer(UD_Utils.
     ID_MENUBUTTON_RFID), new Integer(UD_Utils.PAGE_ID_RFID_INVENTORY)
     );
```

Hierdurch wird eine PageID angelegt, mit einem freien Button verknüpft und dieser passend umbenannt. Nicht vorhanden im FAP Demonstrator sind andersfarbige Buttons zur Anzeige von fehlerhaften Equipment. Diese werden durch Java Swing erstellte, im Vordergrund liegende, Buttontexturen erzeugt. Ebenfalls wird die Anzeige der Detailinformationen zu der Ausrüstung nach anwählen der Buttons mit einem Swing Overlay bewerkstelligt. Das Ergebnis zusammen mit mehreren eingescannten Schwimmwesten ist in Abbildung 3.9 zu sehen.

3.4.2. Programmierung der Steuerlogik im FAP

Um den Buttons die nicht mit Swing Overlays reagieren Funktion zu geben wurde eine Kopie der Klasse: „com.kidsysteme.services.application.usabilitydemonstrator.SeatSettingsApplication.java“ verwendet. Diese Klasse wird beim Aufruf der Seite aktiv, registriert die ihr zugehörigen Buttons und sich selbst mit dem Eventbus und wartet auf Events um Aktionen auszuführen. In diesem Fall das Hinzufügen eines neuen Log Eintrags in das DCL oder den Befehl an das RFID Lesegerät fehlende Tags zu suchen.

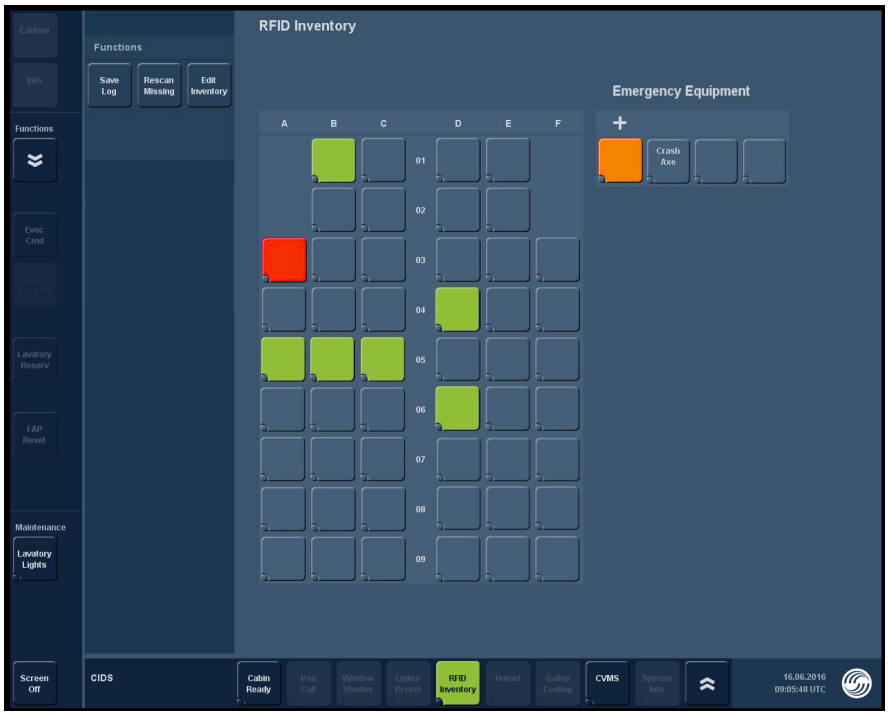

Abbildung 3.9.: RFID Inventory Seite im FAP Demonstrator

3.4.3. Programmierung des Kommunikationsmoduls auf FAP Seite

Wie schon bei dem RFID Lesegerät ist auch beim FAP die WLAN Kommunikation abhängig von dem System und nicht vom FAP Programm. Im FAP Demonstrator wird in der Main Methode ein zusätzlicher Thread gestartet in dem der WSDL Server läuft. Dieser Server implementiert das Interface aus Abbildung 3.2:

```
1  import javax.jws.WebService;
2
3  @WebService(endpointInterface = "de.behrens.abschlussarbeit.
       rfidReaderWebservice")
4  public class rfidReaderWebserviceImpl implements
       rfidReaderWebservice{
5
6          public boolean ping(){
7                  System.out.println("Ping");
8                  return true;
9          }
10
```

```
11          public boolean transferTags(String Tags) {
12                  System.out.println("Tag gelesen: "+Tags);
13                  rfidFAPpage.getInstance().showPageScan(Tags);//Call
                        Swing for Button Overlays.
14                  return true;
15          }
16
17          public boolean needsRescan(){
18                  return rfidFAPpage.getInstance().needsRescan();
19          }
20
21          public String[] rescan(){
22          if(rfidFAPpage.getInstance().needsRescan())
23          return rfidFAPpage.getInstance().getMissingTags();
24                  return new String[]{"",""};
25
26
27          }
28
29  }
```

```
1  import javax.xml.ws.Endpoint;
2
3  public class rfidReaderWebservicePublisher {
4          public static void startWebservice() {
5                  System.out.println("Start webservice");
6                     Endpoint.publish("http://192.168.173.1:9999/ws/
                          rfid", new rfidReaderWebserviceImpl());
7              }
8  }
```

Dieser WSDL Webservice wird durch diesen Code unter „http://192.168.173.1:9999/ws/rfid“ bereitgestellt, was die lokalen WLAN IP Adresse des Demonstrators entspricht. In anderen Hardwareumgebungen müssen IP Adresse und Port gegebenenfalls geändert werden. Dieser Webservice Server kann von Visual Studio zur Programmierung des RFID Lesegerätes referenziert werden und schließlich im Tag Inventarisierungsvorgang verwendet werden. Der Server selbst greift nur auf Methoden anderer Klassen zu und besitzt selbst keine Logik.

3.4.4. Logging im FAP

Der FAP Demonstrator besitzt bereits eine Loggingfunktion zum Speichern von Vorkommnissen in der Kabine. Diese ist auf der Unterseite DCL zu finden. Zum Erstellen von Logeinträgen ist es daher nur notwendig einen neuen Eintrag im vorhandenen Log einzutragen. Die Klasse zur Verwaltung des DCL ist unter „com.kidsysteme.services.application.usabilitydemonstrator.controls“ zu finden. Gespeichert werden die Einträge in dem Attribut „storedDefectsModel“ dem ein neues Objekt der Klasse „DCLDefectModel“ hinzugefügt werden muss. Dieses Objekt speichert die Daten über den aktuellen Inventarisierungsvorgang und kann nun auf der Seite „DCL“, wie in Abbildung 3.10 zu sehen, gefunden werden.

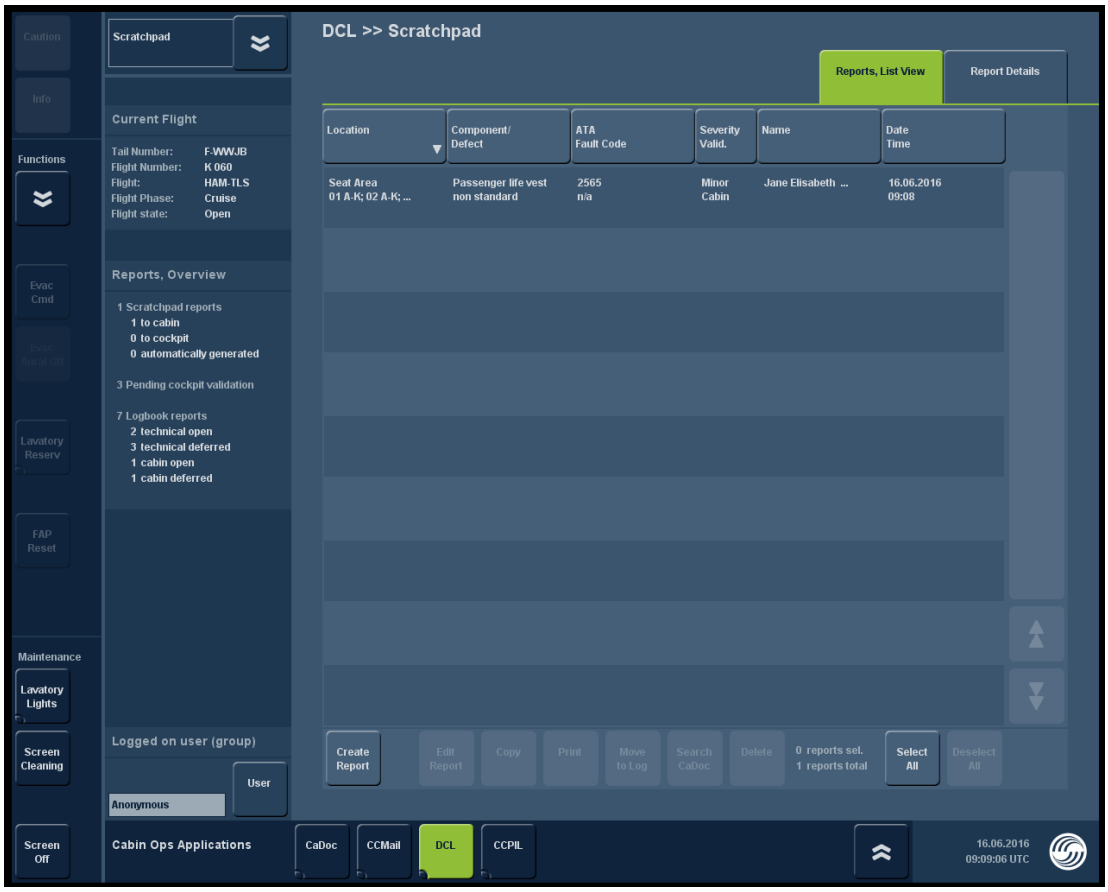

Abbildung 3.10.: FAP Seite des Digital Cabin Logbooks

3.5. Testen

Um einen sicheren Betrieb neuer Software zu gewährleisten, ist es nicht nur wichtig strukturiert zu programmieren, es müssen außerdem abschließend umfangreiche Tests durchgeführt werden um Programmierfehler auszuschließen und eventuelle Planungsversäumnisse aufzudecken. Tests können sowohl für einzelne Funktionen als auch für das vollständige System vorgenommen werden. Kleinere Tests finden parallel zur Entwicklung statt um möglichst frühzeitig Fehlentwicklungen zu entdecken und korrigieren zu können.

3.5.1. Testen des regulären Ablaufes

In Abbildung 3.11 ist der Versuchs- und Demonstrationsaufbau des vollständigen Systems zu sehen. Dieser steht in den Räumes des FKS im Hamburgs Zentrum für Angewandte Luftfahrtforschung (ZAL). Teil des Aufbaus sind das Nordic ID Morphic Lesegerät, ein Monitor mit Touchsteuerung, mehrere RFID Tags sowohl in Schwimmweste verbaut als auch lose und der FAP Demonstrator Rechner. In diesem speziellen Rechner befinden sich zwei getrennte Systeme, die sich nur Ein- und Ausgaben teilen, sonst aber keinen weiteren Datenaustausch untereinander haben. Auf diese Weise wird das Sicherheitskonzept im Flugzeug simuliert. Der FAP Demonstrator Rechner verfügt über eingebautes WLAN welches sich als Accesspoint für das RFID Lesegerät einrichten lässt. Dafür müssen in einer Konsole im Administratormodus folgende Befehle eingegeben werden.

```
1 netsh wlan set hostednetwork mode=allow
2 netsh wlan set hostednetwork ssid=rfid key=rfidtestpw keyUsage=
     persistent
3 netsh wlan start hostednetwork
```

Nach der Eingabe dieser Befehle startet ein Accesspoint mit einem WPA2 verschlüsselten Netzwerk, das den Namen „rfid“ trägt und das Password „rfidtestpw“ besitzt. Mit diesem Netzwerk wird das RFID Lesegerät verbunden. Dabei muss darauf geachtet werden das die IPs, welche vom WLAN vergeben werden, mit den Einstellungen beider Programme übereinstimmt. In diesem Fall sind diese Adressen „192.168.173.1“ für den FAP Demonstrator und „192.168.173.100“ für das RFID Lesegerät. Nach dem Abschluss des Einrichtens können die Programme auf beiden Geräten gestartet werden, um mit den Tests zu beginnen.

Zum Starten des Ablaufes wird auf dem RFID Lesegerät „Inventory“ aufgerufen und der Scan mit „Start“ begonnen. Nun wird das Lesegerät an allen vorhandenen Tags vorbeigeführt, sodass

diese auf dem Display als erkannt angezeigt werden. Im Testaufbau befinden sich sieben Tags, von denen jeweils eines wenige Tage vor dem Verfallsdatum steht und eines darüber hinaus ist. Diese beiden Tags werden auf dem Display in Orange beziehungsweise Rot angezeigt. Alle gescannten Tags werden zeitgleich auch an das FAP übertragen wo, wenn die „RFID Inventory“ Seite ausgewählt ist, die selben Positionen zu sehen sind wie auf dem Lesegerät. Nach dem alle Tags erfasst wurden, wird das RFID Lesegerät durch Druck auf „Send“ den Inventarisierungsvorgang beenden und ein letztes mal alle erfassten Tags zusammen übertragen für den Fall das welche verloren gegangen sind. Nun kann auf dem FAP das Ergebnis betrachtet werden und in dem Fall das es keine weiteren Beanstandungen gibt in das Logbuch übertragen werden. Der erstelle Logeintrag kann nun auf der Seite „DCL“ betrachtet werden. Die FAP Seite wird beim Erstellen des Logs automatisch zurückgesetzt, während auf dem Lesegerät für ein vollständiges Zurücksetzen in den Ausgangszustand auf das Hauptmenü zurückgekehrt werden muss.

Dieser Aufbau und Ablauf wurde bereits auf dem Innovation Summit am 30.05.2016 im ZAL vorgestellt.

Abbildung 3.11.: Demonstrationsaufbau für Versuchs- und Demonstrationszwecke

3.5.2. Testen des Verhaltens bei einem Fehler

Beim Testen geht es allerdings besonders darum das Verhalten der Software in Situationen, die außerhalb der geplanten Abläufe sind festzustellen. Mögliche Fehlerfälle sind hier Probleme

mit den Funkverbindungen oder Bedienungsabläufe, die zwar möglich, aber bei der Planung nicht vorgesehen waren.

Beim Testen des Lesevorganges zeigt sich besonders das die Position der Tags zum Lesegerät sehr wichtig ist. Ein Einscannen der Tags durch einen Stuhl hindurch lässt sich selbst mit maximaler Sendestärke des RFID Lesegerätes nicht einwandfrei gewährleisten. Dies lässt sich möglicherweise durch ein besseres Lesegerät oder bessere Tags lösen, aber mit gegebener Hardware ist es notwendig möglichst gut einscanbare Positionen für die mit Tags ausgestattete Notfallausrüstung zu wählen oder das Lesegerät näher an die Tags zu führen, was allerdings nicht dem Ziel entspricht. Außerdem kommt es vor, dass die WLAN-Verbindung zeitweise unterbrochen wird. Da die Scanergebnisse weiterhin im Lesegerät gespeichert werden und mit einen Druck auf „Send“ ein neuer Übertragungsversuch gestartet wird, ist es möglich eine Position mit besserer Funkverbindung einzunehmen.

Auf der Seite des FAP zeigt sich, dass die Oberfläche nach verlassen und zurückgehen zur „RFID Inventory“ Seite zurückgesetzt wird. Durch eine weitere Übertragung des Lesegerätes stellt die Anzeige wieder korrekt her. In dem Fall das Ausrüstung fehlt oder abgelaufen ist, ist es derzeitig nicht möglich diese im laufenden Vorgang zu ersetzen. Der Inventarisierungsvorgang muss beendet werden, die gespeicherten Ausrüstungsgegenstände ausgetauscht und ein neuer Vorgang gestartet werden. Ein weiterer Fehler, der auftreten kann, ist dass das Lesegerät Verzögerungen bei der Darstellung der Oberfläche zeigt. Der Grund für diesen Fehler konnte noch nicht bestimmt werden. Möglich wären zu viele offene Netzwerkverbindungen, eine ineffiziente Erfassung von zu vielen Tags zeitgleich oder nicht ausreichend leistungsstarke Hardware.

4. Fazit und Ausblick

In dieser Arbeit ging es darum, Notfallausrüstung mit Hilfe eines RFID Systems einzulesen und zur besseren Verwaltung sowie Speicherung auf das FAP zu übertragen. In Studien und Lösungen ohne Integration in das CIDS wurde bereits gezeigt das durch die Nutzung von RFID Zeitersparnisse von bis zu 90 % zu erreichen sind. Die Integration in das CIDS optimiert diesen Vorgang weiter. Nach einer umfangreichen Einarbeitung in die gegebene Ausgangssituation mit existierenden Lösungen und den Vorschriften und Normen für RFID und Flugfahrtindustrie wurde ein Überblick über die zu nutzende Hard- und Software wie dem Nordic ID Morphic und dem FAP Demonstrator verschafft. Der erste Arbeitsschritt nach der Informationsbeschaffung war das erstellen von modellbasierten Anforderungen und den Entwurf von Architekturen bei der Nutzung von UML.

Für das Nordic ID Morphic Lesegerät wurde ein Programm erstellt zum Einlesen von Tags an Notfallausrüstung. Die eingelesen Tags werden mit einer Datenbank verglichen und bei korrekten EPC auf der Oberfläche, die nach FAP Styleguide entworfen wurde, an ihrem zugewiesenen Platz angezeigt. Diese Tags überträgt das Lesegerät dann über WLAN und WSDL zu dem FAP.

Auf dem FAP wurde eine neue Menüseite hinzugefügt, welche die vom Lesegerät geschickten Daten anzeigt. Für den Empfang dieser Daten wurde ein WSDL basierter Webservice Server programmiert mit dem das RFID Lesegerät kommuniziert. Es wird Ausrüstung, die Wartungsbedarf hat auf der FAP Seite angezeigt und es ist möglich genauere Informationen darüber aufzurufen. Der Vorgang wird abgeschlossen, indem aus den gescannten Tags ein Logbucheintag im DCL erstellt wird. Am Ende wurden Tests durchgeführt, mit denen ein vollständiger Inventarisierungsvorgang möglich war und ein partieller „Cabin-Ready“-Status erstellt werden kann. Doch wurden dabei noch einige Schwachpunkte gefunden die beachtet werden müssen.

Besonders die Erweiterung des FAP Demonstrators hat eine große Herausforderung bedeutet. Das Erlangen eines Verständnisses über die Architektur und Funktionsweisen hat viel Zeit gekostet, weshalb einige Funktionen am Ende durch ein Java Swing Overlay abgenommen wurden. Auch hat sich gezeigt, dass der Einsatz der Datenbank nicht so umfangreich notwendig war wie anfangs geplant. Am Ende des Projektes gab es noch den Versuch ein größeres Lesegerät, des Nordic ID Medea zu nutzen. Dieser hätte durch seinen größeren Touchscreen einige Bedienungsprobleme gelöst, wurde aber aus Zeitgründen nicht mehr eingesetzt. Erste Tests haben gezeigt das Betriebssystem und APIs kompatibel sind und es nur einer Anpassung der Benutzeroberfläche auf die größere Auflösung benötigt.

Ausblick

Obwohl die Nutzung im Demonstrationssystem prinzipiell funktioniert, gibt es doch für den Einsatz im Flugzeug noch einige Probleme zu lösen. Besonders die Anbindung über WLan an das CIDS wird im fliegenden System aus Sicherheitsbedenken unwahrscheinlich. Hier müssen Lösungen gefunden werden einen Datentransfer zwischen Lesegerät und Director herzustellen, ohne dabei Sicherheitslücken zu verursachen. Doch das hohe Einsparpotenzial und die sich auch in der Flutfahrt immer weiter durchsetzende RFID-Technologie zeigen das es sich hier um ein Thema handelt das sich lohnt in Zukunft weiter voranzutreiben.

A. Anhang

RFIDInventory.m4 zur Generierung der neuen FAP Seite

```
Page ID 0x13013461

    #
    # general settings (left side)
    #

    # chime inhibit panel
    GROUPING_PANEL(OFFSET_GENERAL_X, OFFSET_GENERAL_Y + 3 * H_GU +
        H_GU_2, 3 * W_GU, 2 * H_GU + H_GU_2, BG_GROUP_PNL_OUT_CA,
        TXT_GROUP_PNL_TITLE_OUT_CA, 3502)
    TOGGLE_BUTTON  ID 0x13003403  BLOCK_GENERAL_PX(0, W_S_BTN)
        BLOCK_GENERAL_PY(4, W_S_BTN)  WI W_BTN  HE H_BTN  BU
        BUTTON_TOGGLE_1x1_IN  SO 0  CO CAP1_EN_BTN_OUT_CA  ENCO
        CAP1_ACT_BTN_IN_CA  ENDA CAP_ACTDIS_BTN_ON_BG_GRP_PNL_OUT_CA
         FO FONT_BUTTON  DA CAP_DIS_BTN_ON_BG_GRP_PNL_OUT_CA  LA 3507
    TOGGLE_BUTTON  ID 0x13013403  BLOCK_GENERAL_PX(1, W_S_BTN)
        BLOCK_GENERAL_PY(4, W_S_BTN)  WI W_BTN  HE H_BTN  BU
        BUTTON_TOGGLE_1x1_IN  SO 0  CO CAP1_EN_BTN_OUT_CA  ENCO
        CAP1_ACT_BTN_IN_CA  ENDA CAP_ACTDIS_BTN_ON_BG_GRP_PNL_OUT_CA
         FO FONT_BUTTON  DA CAP_DIS_BTN_ON_BG_GRP_PNL_OUT_CA  LA 3506
    TOGGLE_BUTTON  ID 0x13023403  BLOCK_GENERAL_PX(2, W_S_BTN)
        BLOCK_GENERAL_PY(4, W_S_BTN)  WI W_BTN  HE H_BTN  BU
        BUTTON_TOGGLE_1x1_IN  SO 0  CO CAP1_EN_BTN_OUT_CA  ENCO
        CAP1_ACT_BTN_IN_CA  ENDA CAP_ACTDIS_BTN_ON_BG_GRP_PNL_OUT_CA
         FO FONT_BUTTON  DA CAP_DIS_BTN_ON_BG_GRP_PNL_OUT_CA  LA 3505

    #
    # content area
    #

```

```
17      # tabs
18      SubPageController  ID 0x13003406  VI 1
19          # pax call
20          Page  ID 0x13013406
21              StaticText BLOCK_CA_PX(0, W_GU_2 + P_BTN)  BLOCK_CA_PY
                    (1, TEXT_3LINE_Y3) CO TXT2_IN_CA  FO FONT_LABEL LA
                    3514
22              GROUPING_PANEL_WITH_FOOTER(OFFSET_CA_X + W_GU_2,
                    OFFSET_CA_Y + 2 * W_GU, CA_WIDTH - W_GU, CA_HEIGHT -
                    2 * W_GU - W_GU_2, BG_GROUP_PNL_IN_CA,
                    TXT_GROUP_PNL_TITLE_IN_CA, 100)
23               # column labels
24              ActiveText ID 0x13003450 BLOCK_CA_PX(1)  BLOCK_CA_PY(2,
                    TEXT_3LINE_Y1) AX 1 FO FONT_HEADER  CO TXT1_IN_CA
25              ActiveText ID 0x13013450 BLOCK_CA_PX(2)  BLOCK_CA_PY(2,
                    TEXT_3LINE_Y1) AX 1 FO FONT_HEADER  CO TXT1_IN_CA
26              ActiveText ID 0x13023450 BLOCK_CA_PX(3)  BLOCK_CA_PY(2,
                    TEXT_3LINE_Y1) AX 1 FO FONT_HEADER  CO TXT1_IN_CA
27              ActiveText ID 0x13033450 BLOCK_CA_PX(4, W_GU_2)
                    BLOCK_CA_PY(2, TEXT_3LINE_Y1) AX 1 FO FONT_HEADER  CO
                     TXT1_IN_CA
28              ActiveText ID 0x13043450 BLOCK_CA_PX(5, W_GU_2)
                    BLOCK_CA_PY(2, TEXT_3LINE_Y1) AX 1 FO FONT_HEADER  CO
                     TXT1_IN_CA
29              ActiveText ID 0x13053450 BLOCK_CA_PX(6, W_GU_2)
                    BLOCK_CA_PY(2, TEXT_3LINE_Y1) AX 1 FO FONT_HEADER  CO
                     TXT1_IN_CA
30              ActiveText ID 0x13063450 BLOCK_CA_PX(7, W_GU_2)
                    BLOCK_CA_PY(2, TEXT_3LINE_Y1) AX 1 FO FONT_HEADER  CO
                     TXT1_IN_CA
31              ActiveText ID 0x13073450 BLOCK_CA_PX(9)  BLOCK_CA_PY(2,
                    TEXT_3LINE_Y1) AX 1 FO FONT_HEADER  CO TXT1_IN_CA
32              ActiveText ID 0x13083450 BLOCK_CA_PX(10)  BLOCK_CA_PY(2,
                     TEXT_3LINE_Y1) AX 1 FO FONT_HEADER  CO TXT1_IN_CA
33              ActiveText ID 0x13093450 BLOCK_CA_PX(11)  BLOCK_CA_PY(2,
                     TEXT_3LINE_Y1) AX 1 FO FONT_HEADER  CO TXT1_IN_CA
34              #temp definition for right and bottom border of the
                    grouping panel
35              pushdef(_RIGHT_BORDER, eval(CA_WIDTH - W_GU - W_GU_2))
36              pushdef(_BOTTOM_BORDER, eval(CA_HEIGHT - H_GU - H_GU_2))
```

```
37              # Scrollbar
38              SCROLLBAR_PAGE_IN(OFFSET_CA_X + 11 * W_GU + W_GU_2,
                    OFFSET_CA_Y + 2 * H_GU + H_GU_2, 9 * H_GU,
                    SCROLLBAR_1x7_IN, 0x13003453, 0x13013453, 0x13023453)
39              #Inhibited Seats text field and keyboardbutton
40              StaticText BLOCK_CA_PX(6)  BLOCK_CA_PY(11, TEXT_3LINE_Y1
                     + H_GU_2) CO TXT2_IN_CA  FO FONT_LABEL LA 3520
41              TEXT_AREA(OFFSET_CA_X + 7 * W_GU + W_GU_2, OFFSET_CA_Y +
                     11 * H_GU + H_GU_2 + H_S_BTN, 4 * W_GU, H_GU - 2 *
                    H_S_BTN, 0x13003456)
42              COMMAND_BUTTON  ID 0x13003457 BLOCK_CA_PX(0,
                    _RIGHT_BORDER + W_S_BTN)  BLOCK_CA_PY(0,
                    _BOTTOM_BORDER + H_S_BTN)  WI W_BTN  HE H_BTN  BU
                    BUTTON_CMD_1x1_IN  IC ICON_KEYBOARD_IN_CA  SO 0
43              #left side buttons (enable all/inhibit all)
44              COMMAND_BUTTON  ID 0x13003454  BLOCK_CA_PX(0, W_GU_2 +
                    W_S_BTN)  BLOCK_CA_PY(0, _BOTTOM_BORDER + H_S_BTN)
                    WI W_BTN  HE H_BTN  BU BUTTON_CMD_1x1_IN  SO 0  CO
                    CAP1_EN_BTN_IN_CA  FO FONT_BUTTON  DA
                    CAP_DIS_BTN_ON_BG_GRP_PNL_IN_CA  LA 3515
45              COMMAND_BUTTON  ID 0x13013454  BLOCK_CA_PX(1, W_GU_2 +
                    W_S_BTN)  BLOCK_CA_PY(0, _BOTTOM_BORDER + H_S_BTN)
                    WI W_BTN  HE H_BTN  BU BUTTON_CMD_1x1_IN  SO 0  CO
                    CAP1_EN_BTN_IN_CA  FO FONT_BUTTON  DA
                    CAP_DIS_BTN_ON_BG_GRP_PNL_IN_CA  LA 3516
46              # status label (num inhibited, num total)
47              StaticText  BLOCK_CA_PX(3)  BLOCK_CA_PY(0,
                    _BOTTOM_BORDER + TEXT_2LINE_Y1)  FO FONT_LABEL  AX 0
                     CO TXT2_IN_CA  LA 3517
48              StaticText  BLOCK_CA_PX(3)  BLOCK_CA_PY(0,
                    _BOTTOM_BORDER + TEXT_2LINE_Y2)  FO FONT_LABEL  AX 0
                     CO TXT2_IN_CA  LA 3518
49              ActiveText  ID 0x13003458  BLOCK_CA_PX(3, -H_S_BTN)
                    BLOCK_CA_PY(0, _BOTTOM_BORDER + TEXT_2LINE_Y1) FO
                    FONT_LABEL  AX 2  CO TXT2_IN_CA
50              ActiveText  ID 0x13013458  BLOCK_CA_PX(3, -H_S_BTN)
                    BLOCK_CA_PY(0, _BOTTOM_BORDER + TEXT_2LINE_Y2) FO
                    FONT_LABEL  AX 2  CO TXT2_IN_CA
51              # multi select button
```

```
52              TOGGLE_BUTTON  ID 0x13003455  BLOCK_CA_PX(4, W_S_BTN)
                    BLOCK_CA_PY(0, _BOTTOM_BORDER + H_S_BTN)  WI W_BTN
                    HE H_BTN  BU BUTTON_TOGGLE_1x1_IN  SO 0  CO
                    CAP1_EN_BTN_IN_CA  FO FONT_BUTTON  DA
                    CAP_DIS_BTN_ON_BG_GRP_PNL_IN_CA  ENCO
                    CAP1_ACT_BTN_IN_CA  ENDA
                    CAP_ACTDIS_BTN_ON_BG_GRP_PNL_IN_CA  LA 3519
53              #
54              # building the seat rows and colums
55              # This must be adjusted if the numer of GUs change
56              #
57              # some temp definitions for simplicity
58              #
59              # param 1 = colum id
60              # param 2 = row id
61              pushdef({"_MAKE_ID"}, {"{"ID 0x13"}$1$2{"345"}"})
62              # param 1 = colum
63              # param 2 = row id
64              # param 3 = row nr
65              # param 4 = opt offset
66              pushdef({"_MAKE_SEAT_BTN"}, {"
67                  TOGGLE_BUTTON  _MAKE_ID($1, $2)2  BLOCK_CA_PX($1,
                        eval($4 + H_GU_2 + W_S_BTN))  BLOCK_CA_PY(eval($3
                         + 2), H_GU_2 + H_S_BTN)  WI W_BTN  HE H_BTN  BU
                        BUTTON_TOGGLE_1x1_IN  SO 0  CO CAP1_EN_BTN_IN_CA
                         FO FONT_BUTTON  DA
                        CAP_DIS_BTN_ON_BG_GRP_PNL_IN_CA  ENCO
                        CAP1_ACT_BTN_IN_CA  ENDA
                        CAP_ACTDIS_BTN_ON_BG_GRP_PNL_IN_CA  LA 100
68                  ActiveImage    _MAKE_ID($1, $2)a  BLOCK_CA_PX($1,
                        eval($4 + H_GU_2 + W_S_BTN))  BLOCK_CA_PY(eval($3
                         + 2), H_GU_2 + H_S_BTN)  WI W_BTN  HE H_BTN  IM
                        WAITING_STATE  MS 16
69              "})
70              # @param 1 = row number id
71              # @param 2 = optional integer row number if row number
                    id > 9 and thus becomes hexadecimal
72              pushdef({"_MAKE_SEATROW"}, {"
73                  ifelse($2,,{"pushdef({"ROWNR"}, eval($1))"},{"
                        pushdef({"ROWNR"}, eval($2))"})dnl
```

```
74                  # create seat colums
75                  _MAKE_SEAT_BTN(0, $1, ROWNR)
76                  _MAKE_SEAT_BTN(1, $1, ROWNR)
77                  _MAKE_SEAT_BTN(2, $1, ROWNR)
78                  _MAKE_SEAT_BTN(3, $1, ROWNR, H_GU_2)
79                  _MAKE_SEAT_BTN(4, $1, ROWNR, H_GU_2)
80                  _MAKE_SEAT_BTN(5, $1, ROWNR, H_GU_2)
81                  _MAKE_SEAT_BTN(6, $1, ROWNR, H_GU_2)
82                  # add row labels
83                  ActiveText  _MAKE_ID(0, $1)1  BLOCK_CA_PX(4, - 2 *
                        W_S_BTN - 3)  BLOCK_CA_PY(eval($1 + 3))  AX 2  AY
                         1  FO FONT_LABEL  CO TXT1_IN_CA
84                  ActiveText  _MAKE_ID(0, $1)1  BLOCK_CA_PX(8, W_GU_2
                        - 2 * W_S_BTN - 3)  BLOCK_CA_PY(eval($1 + 3))  AX
                         2  AY 1  FO FONT_LABEL  CO TXT1_IN_CA
85                  popdef({"ROWNR"})dnl
86              "})
87              _MAKE_SEATROW(0)
88              _MAKE_SEATROW(1)
89              _MAKE_SEATROW(2)
90              _MAKE_SEATROW(3)
91              _MAKE_SEATROW(4)
92              _MAKE_SEATROW(5)
93              _MAKE_SEATROW(6)
94          EndPage
95      EndSubPageController
96          pushdef({"_MAKE_SEATROW_DIALOG"}, {"
97              ifelse($2,,{"pushdef({"ROWNR"}, eval($1))"},{"pushdef({"
                    ROWNR"}, eval($2))"})dnl
98              # create seat colums
99              _MAKE_SEAT_BTN_DIALOG(0, $1, ROWNR)
100             _MAKE_SEAT_BTN_DIALOG(1, $1, ROWNR)
101             _MAKE_SEAT_BTN_DIALOG(2, $1, ROWNR)
102             _MAKE_SEAT_BTN_DIALOG(3, $1, ROWNR, H_GU_2)
103             _MAKE_SEAT_BTN_DIALOG(4, $1, ROWNR, H_GU_2)
104             _MAKE_SEAT_BTN_DIALOG(5, $1, ROWNR, H_GU_2)
105             _MAKE_SEAT_BTN_DIALOG(6, $1, ROWNR, H_GU_2)
106             # add row labels
```

```
107             ActiveText  _MAKE_ID_DIALOG(0, $1)23  BLOCK_CA_PX(2,
                    W_GU_2 - 2 * W_S_BTN - 3)  BLOCK_CA_PY(eval($1 + 2))
                     AX 2  AY 1  FO FONT_LABEL  CO TXT1_IN_CA
108             ActiveText  _MAKE_ID_DIALOG(0, $1)23  BLOCK_CA_PX(7, - 2
                     * W_S_BTN - 3)  BLOCK_CA_PY(eval($1 + 2))  AX 2  AY
                    1  FO FONT_LABEL  CO TXT1_IN_CA
109             popdef({"ROWNR"})dnl
110         "})
111         _MAKE_SEATROW_DIALOG(0)
112         _MAKE_SEATROW_DIALOG(1)
113         _MAKE_SEATROW_DIALOG(2)
114         _MAKE_SEATROW_DIALOG(3)
115         _MAKE_SEATROW_DIALOG(4)
116         _MAKE_SEATROW_DIALOG(5)
117         _MAKE_SEATROW_DIALOG(6)
118     EndPage
119
120     Page  ID 0x12003459
121         pushdef({"_DIALOG_LEFT_BORDER"}, eval(OFFSET_GENERAL_X + (
                GENERAL_WIDTH + CA_WIDTH - 6 * W_GU) / 2))
122         pushdef({"_DIALOG_TOP_BORDER"}, eval(OFFSET_GENERAL_Y + (
                CA_HEIGHT - 2 * H_GU + H_GU_2) / 2 - H_GU_2))
123         # dimming background
124         DIMMING_GENERAL_CA
125         DIALOG_INFO(_DIALOG_LEFT_BORDER, _DIALOG_TOP_BORDER, 6 *
                W_GU, 2 * H_GU + H_GU_2, BG_CA, 3521, 0x12013459, 27, 0
                x12003459, 26)
126         # cushion text
127         StaticText  PX eval(_DIALOG_LEFT_BORDER + P_BTN)
                BLOCK_CA_PY(0, _DIALOG_TOP_BORDER + H_GU_2) CO TXT1_IN_CA
                  FO FONT_LABEL  LA 3522
128     EndPage
129 EndPage
130 # Pop local definitions
131 popdef({"_DIALOG_TOP_BORDER"})
132 popdef({"_DIALOG_LEFT_BORDER"})
133 popdef({"_MAKE_SEATROW_DIALOG"})
134 popdef({"_MAKE_SEAT_BTN_DIALOG"})
135 popdef({"_MAKE_ID_DIALOG"})
136 popdef({"_INPUT_WIDTH"})
```

```
137 popdef({"_BOTTOM_BORDER"})
138 popdef({"_RIGHT_BORDER"})
139 popdef({"_MAKE_ID"})
140 popdef({"_MAKE_SEAT_BTN"})
141 popdef({"_MAKE_SEATROW"})
```

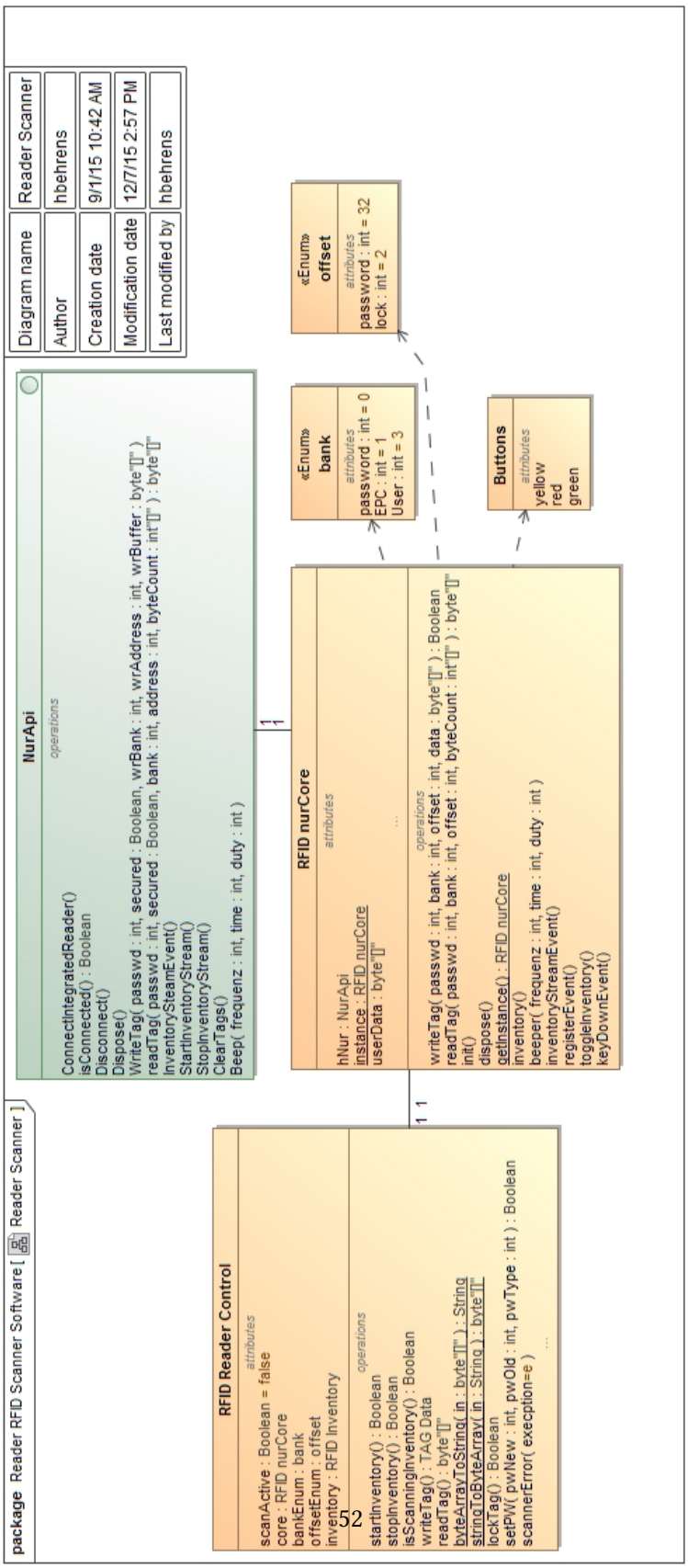

Abbildung A.1.: Klassendiagramm das des den RFID LesegerÃ¤tes steuernden Programms

Literaturverzeichnis

[ant16] *ANT.* http://www.eclipse.org/eclipse/ant/, 2016. – Zugriffsdatum: 08.06.2016

[BOA15] *Lufthansa Airbus A320-214 (D-AIZN) Boarding at Paris [LFPG] Charles de Gaulle, France* . https://youtu.be/oMm-oRVjVqE?t=38s, 2015. – Zugriffsdatum: 08.06.2016

[CID16] *Logischer Aufbau des A320 CIDS.* http://aviation.stackexchange.com/questions/2699/what-can-the-flight-attendant-panel-do, 2016. – Zugriffsdatum: 08.06.2016

[CIO06] *CIO: Funkt im RFID-Markt.* http://www.cio.de/knowledgecenter/scm/827357/index.html, 2006. – Zugriffsdatum: 16.03.2016

[CWR06] *Computerwoche: RFID-Boom hat gerade erst begonnen.* http://www.computerwoche.de/nachrichten/579177/index.html, 2006. – Zugriffsdatum: 16.03.2016

[ecl16] *Eclipse.* https://eclipse.org/, 2016. – Zugriffsdatum: 08.06.2016

[EPC08] *Specification for RFID Air Interface.* Oktober 2008. – Grundlage des ISO18000-6

[GB06] Glover, Bill ; Bhatt, Himanshu: *RFID essentials.* 1. Sebastopol, CA : O'Reilly, 2006. – ISBN 978-0-596-00944-1

[GO115] *PR Newswire: Global RFID Market 2015-2020.* http://www.prnewswire.com/news-releases/global-rfid-market-2015-2020---readers-tags-software-and-servic
html, 2015. – Zugriffsdatum: 16.03.2016

[IMG03] *How to Create a Microsoft .NET Compact Framework-based Image Button.* https://msdn.microsoft.com/en-us/library/aa446518.aspx, 2003. – Zugriffsdatum: 08.06.2016

[ISO15] *Information technology Radio frequency identification for item management.* Oktober 2015. – Informationstechnik - Identifizierung von Waren mittels Hochfrequenz (RFID) für das Management des Warenflusses

[JAV16] *Java.* https://www.java.com/de/, 2016. – Zugriffsdatum: 08.06.2016

[MME16] *Master Minimum Equipment List.* http://fsims.faa.gov/wdocs/mmel/a-350_rev0.pdf, 2016. – Zugriffsdatum: 02.06.2016

[nor16] *Nordic ID Products.* http://www.nordicid.com/en/home/products/mobile-readers/, 2016. – Zugriffsdatum: 10.06.2016

[RFA12] *Airbus weitet den Einsatz von RFID zur Teileidentifikation auf all seine Flugzeugfamilien aus.* http://www.rfid-ready.de/201211132949/airbus-weitet-den-einsatz-von-rfid-zur-teileidentifikation-auf-html, 2012. – Zugriffsdatum: 16.03.2016

[RFI13] *Schwimmwestenkontrolle mit Hilfe von RFID.* http://www.lufthansa-technik.com/rfid, 2013. – Zugriffsdatum: 08.06.2016

[RFI14] *RFID Tag Funktionsweise.* https://www.barcodesinc.com/info/buying-guides/rfid.htm, 2014. – Zugriffsdatum: 08.06.2016

[RFI16] *RFIDTAGS.* http://www.rfid-webshop.com/index.php/cat/c1_Transponder.html/XTCsid/15dd, 2016. – Zugriffsdatum: 08.06.2016

[RFJ15] *RFID-Journal.* http://www.rfid-journal.de, 2015. – Zugriffsdatum: 16.03.2016

[RFT07] *RFID-Technologie.* https://de.wikibooks.org/wiki/RFID-Technologie, 2007. – Zugriffsdatum: 16.03.2016

[RJM12] *Minimum Equimpent List der Royal Jordanian für Airbus der A320 Familie.* https://de.scribd.com/doc/160517091/A319-A320-A321-MEL-Sep12, 2012. – Zugriffsdatum: 02.06.2016

[SAE06] *Passive RFID Tags Intended for Aircraft Use.* Dezember 2006. – Leitfaden für passiven UHF RFID-Transponders mit Luft- und Raumfahrtzulassung

[si206] *Silicon: RFID Wachstum 2007.* http://www.silicon.de/39180208/marktforscher-sieht-2007-weniger-rfid-wachstum/, 2006. – Zugriffsdatum: 16.03.2016

[Spe09] *ATA Spec2000 Chapter 9 - Automated Identification and Data Capture.* Juni 2009. – Techniken und Standards die weltweit vom Großteil der Luftfahrtindustrie akzeptiert sind

[STU10] *Thomas Cook Life Vest Case Study.* http://www.corerfid.com/Files/Case%20Studies/106%20Thomas%20Cook%20Life%20Vest%20Case%20Study.pdf, 2010. – Zugriffsdatum: 08.06.2016

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 17. Juni 2016 Hendrik Behrens